Universidade de Aveiro Departamento de Física
2010

André Machado Lindo

Detecção de elementos estranhos em modelos inspirados em imunologia

Nonself detection in immune-inspired models

Universidade de Aveiro Departamento de Física
2010

André Machado Lindo

Detecção de elementos estranhos em modelos inspirados em imunologia

Nonself detection in immune-inspired models

dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Física, realizada sob a orientação científica do Dr. Fernão Rodrigues Vístulo de Abreu, Professor auxiliar do Departamento de Física da Universidade de Aveiro

o júri

Presidente	Prof. Dr. João Filipe Calapez de Albuquerque Veloso professor auxiliar do Departamento de Física da Universidade de Aveiro
Orientador	Prof. Dr. Fernão Rodrigues Vístulo de Abreu professor auxiliar do Departamento de Física da Universidade de Aveiro
Arguente	Prof. Dr. Bruno Miguel Paz Mendes de Oliveira professor auxiliar da Faculdade de Ciências da Nutrição e Alimentação da Universidade do Porto

acknowledgements

To my parents, for raising me and supporting my academic studies.

To my project supervisor, Professor Fernão Abreu, for his intellect, for his guidance and for what I've learned from him.

To Bruno Faria, Patricia Mostardinha, and my other colleagues of the Bio-Inspired Physics Group, for the help and useful discussions.

To my closest friends.

palavras-chave

sistemas imunológicos artificiais, selecção negativa, frustração celular, discriminação *self* / *nonself*.

resumo

Neste trabalho é apresentado um algoritmo para detecção de elementos estranhos (*nonself*) baseado no mecanismo de Frustração Celular. Este mecanismo apresenta uma nova abordagem às interacções celulares que ocorrem no sistema imunológico adaptativo. O conceito é o de que qualquer elemento estranho estabelecerá interacções menos frustradas do que os restantes elementos do sistema, podendo por isso, através do seu comportamento anómalo, ser detectado. O algoritmo proposto possui vantagens em relação aos sistemas imunológicos artificiais mais conhecidos. Entre elas está a possibilidade de obter detecção perfeita com um número reduzido de detectores. Nesta tese, analisa-se comparativamente este algoritmo com algoritmos de selecção negativa existentes na literatura.

keywords artificial immune systems, negative selection, cellular frustration, self/nonself discrimination.

abstract In this work an algorithm for nonself detection is presented, based on the Cellular Frustration mechanism. This mechanism presents a novel approach to cellular interactions occurring in the adaptive immune system. The concept is that any nonself element will establish less frustrated interactions than the remaining elements of the system, can thus, by its anomalous behaviour, be detected. The proposed algorithm has advantages over the most know artificial immune systems. Among the advantages there is the possibility to achieve perfect detection using a reduced number of detectors. In this thesis, this algorithm is analysed comparatively to negative selection algorithms that can be found in literature.

“There is nothing worse for mortals
than a wandering life.”

Homer, *Odyssey*

List of Figures

List of Tables

Contents

Introduction

It is impractical to find and patch every security hole in a large computer system. The need for more comprehensive approaches to security is constantly increasing due to the potential space for a malicious code. This potential space grows exponentially with the number of instructions, and so, it is virtually infinite. Classical protection mechanisms mostly use detection of known signatures or complete mapping of every potential harmful signature (positive approach). On one hand, this approach requires large databases to store all the known harmful signatures. It is also unpractical to verify all the signatures in such a large database. On the other hand, there is the problem of mimicry by malicious codes (viral) which are similar to known safe codes (non-viral). This would even require scanning all the surroundings to verify very similar codes against the database. This kind of approaches is also not prepared to novel types of intrusions, i.e., unknown codes.

The problems presented suggest a discussion whether to use negative approaches instead of a positive approach (as currently used in computer antivirus). The negative approaches use the known safe information (*self*) to detect completely unknown viral codes (*nonself*). In this case, the known information is quite less in size and diversity. The use of immune-inspired algorithms is a pertinent solution in this context. The immune system provides a robust and multi-layered protection of the human body against external agents (*nonself*). Artificial immune systems (AIS) draw inspiration from the immune system's specific properties such as recognition, learning and memory.

Stephanie Forrest and collaborators built approaches using the so-called Negative Selection Algorithms (NSA). NSAs are immune-inspired algorithms dealing with *nonself* detection in data of computer systems. These algorithms consist in generating the smallest set of detectors to cover the largest possible regions of the *nonself* space. However, in this kind of approaches, achieving perfect nonself detection is unachievable. NSAs require an unreasonable number of detectors to obtain good detection performances. To overcome this problem and reduce the size of the detector set, the detection performance considered is safe (~0.9) but not perfect detection.

A different approach is proposed in this thesis to address the same motivation as NSAs. The mechanism developed by the supervisor of this thesis is based on the Cellular Frustration framework. Pathogenic interactions will be less frustrated than the interactions of the remaining system and, therefore, because of its anomalous behaviour can be detected. In the literature, we have outlined the possibility of using cellular frustration approaches to discuss immunological mechanisms. It has been

explained, for instance, that it is possible to conceive the existence of tolerance with completely reactive cells. Conventionally, immunologists do not think that it is possible that very reactive cells can achieve tolerance in this way of cells' frustration. It has been discussed how to reconcile these two apparent incompatible concepts, which is: high reactivity against anything that was not previously in the system (*nonself*) and complete tolerance against everything that has already been presented (*self*).

The main contribution of this thesis concerns the formulation and validation of a Cellular Frustration algorithm (CFA) for self/nonself discrimination in general data sets. This work follows from the PhD work under development by Patricia Mostardinha. In her work she recently showed, using the Interaction Lists (ILs) formalism, that it is possible to define a Cellular Frustration algorithm to perform perfect self/nonself discrimination on a set of arbitrarily diverse elements (this work is in preparation for publication). Important issues remained on how these results could be applied on practical settings as, for instance, those addressed by Forrest and collaborators, where an intruder is looked for in a data set made of binary strings. Mostardinha's approach defines T cells receptors as Interaction Lists, and the similarity between ligands as ranks in these Interaction Lists. Furthermore, the number of detectors considered in this approach is of the size of the self set, which is completely opposite to the approach of Forrest and collaborators. The aim of this thesis is to define algorithms that use only strings to code the information Mostardinha's put on Interaction Lists. A huge information compression takes place in this passage. Consequently, it becomes nontrivial to understand whether such a class of algorithms could perform as well as predicted by Mostardinha by more theoretical arguments. We also want to compare the detection performances of the developed algorithm with results from NSAs found in literature.

This thesis is organized as follows.

In chapter 1, Negative Selection algorithms (NSA) are presented, introducing the immunological concepts that inspired this kind of algorithms. The algorithmic approaches proposed by Forrest and collaborators for nonself detection are explored and results for diverse cases are obtained with numerical simulations.

In chapter 2, the Cellular Frustration framework is discussed. The mechanism is explained with some examples and different possible models. A new model for nonself detection is presented, providing the basis for the Cellular Frustration algorithm developed in chapter 4. Comparisons with the approaches by Forrest and collaborators are established. This approach led to the development of strategies based on very diverse coding binary rules to compress the information contained in Interaction Lists. In chapter 3, sets of binary rules are explored, focusing their diversity and application in the Cellular Frustration algorithm developed in chapter 4.

In chapter 4, a Cellular Frustration algorithm is developed for nonself detection using binary representation. The algorithm's features are explained and some improvements introduced. Results from numerical simulations are presented for diverse systems and compared with results of Negative Selection algorithms. This thesis ends with overall conclusions.

Chapter 1

Negative Selection Algorithms

Negative Selection algorithms (NSA) are immune-inspired algorithms developed for complex anomaly detection problems. This type of algorithm tries to mimic the process of self/nonself discrimination that results from the negative selection mechanism in the thymus.

This chapter focuses on NSAs developed by Forrest and collaborators [1-3]. First I will introduce some basic immunological concepts needed to understand negative selection in the immune system. Afterwards I show how computational algorithms were developed inspired by these ideas. Important mathematical quantities used to measure the distance between two strings are defined. Results from numerical simulations are presented and detection performances examined for several possible parameters. These results are important for a comparison with the algorithm we are developing which is introduced in chapter 4.

1.1 Basic Immunology

The human body is constantly subject to attacks from a huge diversity of micro-organisms such as bacteria, parasites, viruses, and fungi, known collectively as pathogens. These pathogens are a source of many disorders. For instance, pneumonia is caused by bacteria, AIDS and influenza are caused by viruses, and malaria is caused by parasites. Pathogens can be especially harmful because they replicate. A limited definition of the Immune System's "purpose" would be that the Immune System exists to protect the body from pathogens, and to do so while minimizing damages inflicted to the body and maintaining its continued functioning. The two main problems that the Immune System faces are:

1. The *detection* of pathogens;
2. The *elimination* of pathogens while minimizing damaging the body.

This thesis deals only with the problem of detecting pathogens (1.). This problem is often described as that of distinguishing "self" from "nonself". That is, distinguishing elements that belong to the body (self) from pathogens (nonself) [4-6].

The Immune System is divided in two main structures: the innate immune system and the adaptive immune system. All vertebrates are born with the *innate* immune system. This system is formed by a set of cells that are able to recognize and bind to common molecular patterns, eliminating the associated pathogens. The innate immune system is inherited from ancestors and is mainly static. Cells from the innate immune system are also used to present molecular patterns to cells from the adaptive immune system and possibly initiating an adaptive immune system response.

The *adaptive* immune system will be focused in this thesis. This system is "antigen-specific", which means it adapts and "learns" to recognize and attack only specific kinds of pathogens. It also retains "memory" for future detections. Another interesting feature is that the primary response to a pathogenic invasion often becomes apparent only a few days after the infection. The adaptive immune system is made up of white blood cells, called lymphocytes. These lymphocytes can be subdivided in two classes: B and T cells. They carry recognition units on their surface, called antibodies and receptors, respectively. A vast majority of researchers believe that through these recognition units the adaptive immune system is capable of performing self/nonself discrimination.

Immature T cells are produced in the bone marrow. In order to perform self/nonself discrimination T cells must undergo a maturation process. This process occurs in the thymus gland in two main phases: *positive selection* and *negative selection*.

Antigen presenting cells (APCs) are cells from the innate immune system that display fragments of antigens to T cells. They present them on their surface through a major histocompatibility complex (MHC). These fragments are called epitopes and constitute the APCs' ligand used as a "binding region" to T cell receptors.

In *positive selection*, T cells are selected depending on whether they are fully functional in recognizing MHC molecules. In this phase, non-functional T cells which are unable to bind to MHC molecules, are eliminated by a process called "death by neglect" [5]. This mechanism is also known as MHC restriction.

Negative selection is the mechanism used by the immune system to avoid that lymphocytes react against self cells (tolerance). The thymus allows thymocytes (immature T cells) to mature in an environment protected from contact with foreign antigens (blood-thymic barrier). During this process antigen presenting cells present only self peptides through MHCs to T cells. T cells binding with high affinity will be eliminated by apoptosis. As a result, only T cells reacting strongly to nonself peptides will remain. Negative selection provides in this way the possibility of distinguishing self antigens from nonself antigens because, in principle, only those T cells that bind (strongly) against any peptide that is not a self peptide will remain.

Only after the maturation process T cells are able to recognize different antigenic peptides presented by antigen presenting cells (APC).

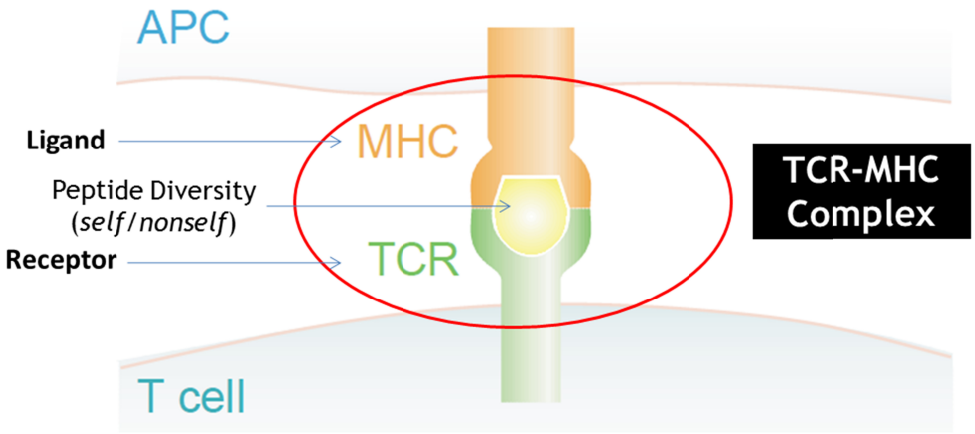

Fig. 1.1 Representation of surface elements of T cells and APCs. The major histocompatibility complex (MHC) of the antigen presenting cell (APC) binds to the T cell receptor (TCR). The MHC contains peptide fragments from the antigen inside within the cell, and is recognized by the T cell. (adapted from [7])

There are some key features of the immune system that are relevant for this thesis. Detection or recognition events occur by ligand-receptor interactions (APC-Tcell), forming the TCR-MHC complex. These interactions are specific and affinity-based. And because recognition is specific, despite that each type of receptor can recognize many different pathogens, there should be a sufficiently diverse repertoire of receptors to cover the whole pathogen space.

The immune system needs to detect and eliminate pathogens as quickly as possible, because pathogens replicate exponentially. The immune system incorporates mechanisms that "learn" and adapt to specific foreign proteins.

Pathogens should be eliminated but, on the other hand, self cells must be preserved in order to protect the body and maintain homeostasis. This is called tolerance. The immune system has mechanisms to avoid autoimmune responses by inducing tolerance to self (e.g., negative selection). Specialized T cells may be responsible for tolerance (e.g., T-helper and T-reg cells) as well as for memory.

1.2 Introduction to Negative Selection Algorithms

The Immune System is a remarkable and complex natural system: it provides both defence and maintenance of the body. Artificial Immune Systems (AIS) (also known as immunological computation) are a field devoted to the study of computational models based on the principles of biological Immune System. The broad use of biologically inspired Artificial Immune Systems is mainly due to its multi-layered defence mechanisms.

In particular, negative selection algorithms (NSA) are anomaly/change detection approaches in Artificial Immune Systems. NSAs try to replicate the negative selection process involved in the maturation of T cells in the thymus. The majority of the algorithms begin by reproducing the random production of T cells in the bone marrow. Then, in the thymus, T cells that bind self peptides are eliminated. By negative selection, only T cells that bind nonself are kept. Only after this process, the selected mature T cells are ready to detect pathogens. Table 1.1 contains a correspondence between a list of the terminologies used in the two systems.

Table 1.1 Biological to computational correspondence of structures.

Immune System	Computer System
T cells, antibodies	detectors
APCs, antigens	presenters
epitopes, peptides	strings of data
pathogens	anomalous data, computer viruses
self antigens	self set
affinity measure	matching rule

The main strategy used by NSAs is to cover the nonself space with a suitable set of detectors. The operation of the majority of NSAs consists of two stages: *censoring* and *monitoring* [5]. The first stage comprehends the detector generation. Strings are generated randomly and compared with a self set. By using a defined matching rule, candidate detectors that match any self string are discarded, and unmatched ones are kept.

The second stage consists in the detection of the nonself. Detectors generated in the first stage are checked continuously against incoming strings. Using the matching rule, strings are classified as nonself if there is a matching, and as self otherwise.

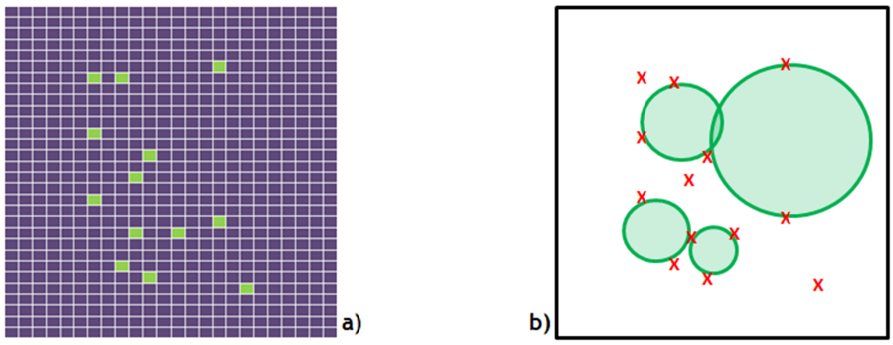

Fig. 1.2 Different representations of self and nonself space. For different anomaly detection problems, domains' space can have abstract representations and different complexity. a) 2-Dimensional possible representation (*self* in green). b) Example of different sized detectors (in green) showing an overlap, in the 2-Dimensional representation (*self* with red crosses).

NSAs work through the definition of the complement set of the self set. The purpose is to discriminate self and nonself giving as an input the set of self samples only. These are called "one-class learning" algorithms. No prior knowledge of nonself is required.

Particular NSAs are characterized by the way detectors are represented, the matching rules used, and the mechanisms for generating detectors and discarding self-reactive candidate detectors.

A generic NSA is presented in pseudo-code as follows:

Algorithm 1.1: Generic Negative Selection Algorithm.

input : $\boldsymbol{S}$ = set of self elements, $\boldsymbol{U}$ = shape space, $\boldsymbol{S} \subset \boldsymbol{U}$
1 **begin**
2 Generate a set $\boldsymbol{R}$ of detectors, such that each fails to match any element in $\boldsymbol{S}$.
3 Monitor data continuously by matching the detectors in $\boldsymbol{R}$ against any string in $\boldsymbol{\delta}$. If any detector matches with $\boldsymbol{\delta}$ consider the string as nonself $\boldsymbol{N}$, otherwise as self $\boldsymbol{S}$.
4 **end**

1.3 Detector Generating Algorithms

Here I will present algorithms proposed by Forrest and collaborators [1-3]. The first algorithm was proposed in 1994 [1] as one of the earliest artificial immune systems and consequently it received a lot of attention. Since then, many variations have been proposed, such as the *linear time* and the *greedy* algorithms [2, 3]. Here I will only discuss the original algorithm (known as the *exhaustive search* algorithm) and the *linear time* algorithm.

An important feature in these algorithms is how distances between strings are established, which is known as matching rules. Matching rules are crucial to the performance of the algorithm. There are many matching rules that can be defined depending on the application or motivation. Examples are such as Hamming distance, *r*-chunk and Rogers-Tanimoto, etc. The matching rule used by Forrest and collaborators is the r-contiguous matching rule, also called "r-contiguous bits" (*rcb*). It is well accepted among theoretical immunologists the suitability of the r-contiguous matching rule to quantify the binding strength between antibodies and antigens [5, 6]. This matching rule works as an affinity threshold between the complementary immunological structures and has a natural representation in binary coding.

Given a matching threshold r, if there are $x < r$ contiguous bits between two strings, there is no matching. Otherwise, if there are at least r contiguous bits $(x \geq r)$, then matching takes place.

Definition A bit string b with $b = b_1 b_2 \ldots b_l$, and a detector d with $d = d_1 d_2 \ldots d_l$, match with r-contiguous rule, if a position p exists where $b_i = d_i$, for $i = p, \ldots, p + r - 1$ and $p \leq l - r + 1$.

The value of r determines the detectors' degree of specificity: the smaller the value of r, the more general is the detector (higher detection coverage). The r-contiguous matching rule relax the matching requirements by diminishing the number of bits that have to be matched (not the whole string). It also enlarges the region covered by each detector.

A perfect match is rare for a reasonable string length l and matching threshold r. The probability of two random strings of length l matching under r-contiguous bits [1], for binary alphabets, depends on the matching threshold r through the equation:

$$P_m \approx 2^{-r}\left(\frac{l-r}{2}+1\right) \tag{1.1}$$

To understand the relationships between the different string domains involved in this kind of algorithms [2, 3], the following relationships are represented in figure 1.3. Consider U the space of all possible strings and U_d the space of all possible detectors. Given a set of self strings $S \subset U$, the purpose of the NSA is to find a detector set $R \subset U_d$ that matches as many of the nonself strings N as possible $(N = U - S)$ without matching any of the self strings in S. The detector strings in R are chosen from the candidate detectors C, consisting of all strings that do not match any string in S. If the detector set R has all the candidate detectors C, it would be possible to match all the detectable nonself strings N'. However, a fraction of the total nonself strings N is misclassified as being self, which is known as holes $(H = N - N')$. Holes appear because no possible detector can match these strings. It should be stressed out that holes are unavoidable in NSAs. The number of holes decreases if the matching threshold r is increased, i.e., having smaller detection regions. However, this is achieved at the expense of having an increasing number of detectors necessary.

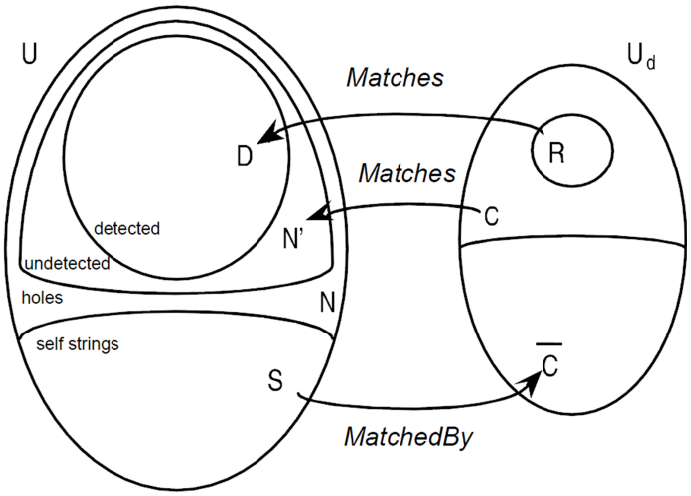

Fig. 1.3 Schematics of string space (U) and detector space (U_d) and their subsets. String space is partitioned into self strings (S) and nonself (N). Candidate detectors (C) are those that do not match any string in S. The detector set R is a subset chosen from C. Nonself strings can be subdivided in detectable nonself strings (N') and holes $(H = N - N',$ not labeled). Detectable nonself strings can be detected (D) or not $(FN = N' - D,$ not labeled), depending on the choice of the detector set R. (from [3])

Generally, only a subset of the candidate detectors is included in R, allowing only the detection of all nonself strings in D. False negatives FN are nonself strings which are not detectable because the detector set R is smaller than the set of all candidate detectors $(FN = N' - D)$. Holes are the leftover undetectable strings after all the possible detectors have been defined. False positives would be self strings misclassified as being nonself. However, negative selection algorithms from Forrest and collaborators do not allow this type of errors: the number of false positives is always equal to 0.

Considering the classification represented in figure 1.3, the following relationships can be established [5, 6]:

$$\boldsymbol{U = S \cup N}, \qquad \boldsymbol{S \cap N = \emptyset} \tag{1.2), (1.3}$$

$$\boldsymbol{D \subseteq N' \subset N} \tag{1.4}$$

$$\boldsymbol{U = S \cup N' \cup H}, \qquad \boldsymbol{N = H \cup N'} \tag{1.5), (1.6}$$

$$\boldsymbol{N' \cap H = \emptyset}, \qquad \boldsymbol{S \cap H = \emptyset} \tag{1.7), (1.8}$$

$$\boldsymbol{R \subseteq C} \tag{1.9}$$

To further analyse these algorithms, some notation and corresponding definitions are introduced:

N_{R_0} = initial detector repertoire size (candidate detectors), before *censoring*
N_R = detector repertoire size, after *censoring*
N_S = number of self strings (size of the self set, $|S|$)
P_m = probability that a randomly chosen string and detector match according to the matching rule
f = probability of a random string not matching any of the N_S self strings
P_f = probability that a single random nonself string will not be matched by any detector in R, i.e., probability of non-detection
N_H = number of holes

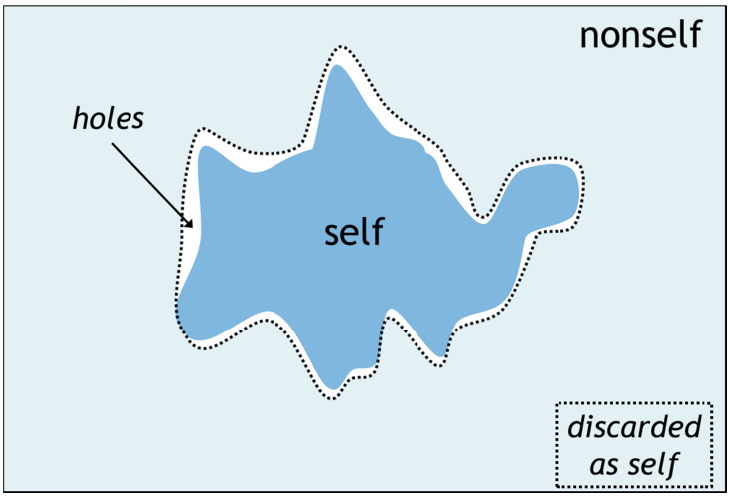

Fig. 1.4 The appearance of holes. Self space will be matched differently depending on the matching threshold r. Holes are created when candidate detectors that match these strings are eliminated due to matching with a self string. Holes are nonself strings that are discarded as matching self, and so, no valid detectors are possible to generate that match these strings.

1.3.1 The Exhaustive Search Algorithm

An exhaustive search algorithm was first proposed by Stephanie Forrest and collaborators in 1994, and it was claimed to be analogous to the natural negative selection process in the thymus [1-4]. The algorithm is similar to Algorithm 1.1 presented above. Candidate detectors d are drawn randomly and checked against all the self strings, using the r-contiguous bits rule. Strings which fail to match the self set S, are kept as valid detectors. The detector set R is obtained by repeating this process until a number of desired detectors is achieved. Nonself detection is achieved by checking every given string with all the generated detectors (one by one). The several steps involved in this algorithm are illustrated in the following figure.

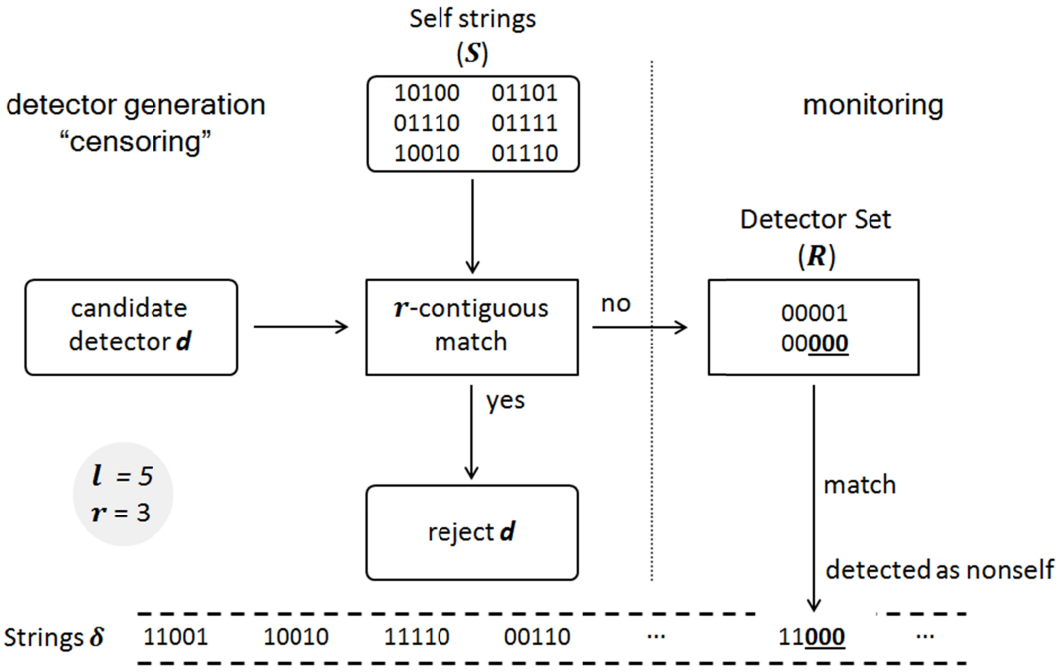

Fig. 1.5 Operation of the NSA using the exhaustive search algorithm. Candidate detectors d that do not match self strings are collected to the detector set R. Then by continuously matching strings with the detector set, positive matchings are considered nonself detections. (adapted from [6])

The algorithm tests N_{R_0} random candidate detectors building a detector set R of size N_R. The probability of matching self (f), is then related by:

$$N_R = N_{R_0} \cdot f \tag{1.10}$$

Here f is:

$$f = (1 - P_m)^{N_S} \tag{1.11}$$

Where P_m is the probability of matching and N_S is the size of the self set.

For small P_m and a large N_S this can be approximated by:

$$f \approx e^{-P_m N_S} \tag{1.12}$$

Similarly, for the probability of failure P_f:

$$P_f = (1 - P_m)^{N_R} \tag{1.13}$$

Which for low P_m and large N_R, can be approximated by:

$$P_f \approx e^{-P_m N_R} \tag{1.14}$$

Using expressions (1.10) and (1.14), it is possible to find that:

$$N_R = N_{R_0} \cdot f \approx \frac{-lnP_f}{P_m} \tag{1.15}$$

Solving (1.15) with respect to N_{R_0}, and then using (1.12), it is easy to obtain:

$$N_{R_0} = \frac{-lnP_f}{P_m \cdot (1 - P_m)^{N_S}} \approx \frac{-lnP_f}{P_m} \cdot e^{P_m N_S} \tag{1.16}$$

With these formulas it is possible to calculate theoretically the size of the initial repertoire N_{R_0} for a fixed failure probability. These formulas are approximate for the case of independent detectors. However, when P_m or N_S increase, detectors become less independent which means that more detectors overlap. This results in a higher failure probability than it would be expected from (1.13). Other considerations concerning these formulas are thoroughly discussed in [1-3].

Relatively to the computation time of this algorithm, the algorithm depends exponentially in the size of N_S as is clear from (1.16). In fact, the algorithm time complexity depends on finding suitable detectors among all candidate detectors. In terms of memory space complexity the algorithm depends only on storing the N_S self strings of length l.

time: $\mathcal{O}\left(\frac{-lnP_f}{P_m \cdot (1 - P_m)^{N_S}} \cdot N_S\right)$

space: $\mathcal{O}(l \cdot N_S)$

This algorithm works with any matching rule. However it has the disadvantage of being inefficient for detector generation. The *linear time* and *greedy* algorithms were developed to be more time efficient.

1.3.2 Linear Time Algorithm

The *linear time* detector generating algorithm was developed in order to improve the computational practicality of the *exhaustive search* approach. The approach considered by Forrest and collaborators considers the counting of candidate detectors with common patterns (templates). This feature avoids selecting similar detectors, which is recurrent in random detector generation. The algorithm requires

the storage of a matrix C that contains information about all the possible ways two strings can match over r-contiguous bits. Furthermore, matrix C contains the number of all correlated strings that are unmatched by the self set S.

The *linear time* algorithm follows a two-phase scheme [2, 3, 5]. In Phase I, the counting recurrence of the candidate detectors set is overcome. In Phase II, a detector set R is generated randomly using the enumeration created previously in Phase I. The following notation is used to describe the method:

$\hat{s}$ – the string s stripped of its first bit.

$\hat{s} \cdot b$ – the string $\hat{s}$ appended with b after the last bit, where $b \epsilon \{0,1\}$

$t_{i,s}$ – template with r fully specified contiguous bits starting at position i given from the string s. The template is a size l string with $l - r$ blank symbols (e.g., $*** 101 *$)

$C_i[s]$ – number of right completions of $t_{i,s}$ unmatched by any string in S

(*example*: with $l = 7, r = 3, s = 110$, $t_{3,s} = ** 110 **$, one right completion for $t_{3,s}$ is $** 11010$)

Phase I: Solving the counting recurrence. Here, the following question is answered: how many l-bit strings containing the template s are not matched by any string in S? This question is answered recursively by calculating $C_i[s]$. This means evaluating all the possible template matchings from $i = l - r + 1$ to 1. Starting from $C_{l-r+1}[s]$, there are no blanks to the right and so, no right completions. Therefore, $C_{l-r+1}[s]$ will be zero if the template $t_{l-r+1,s}$ is matched in S, one otherwise. For all the other $C_i[s]$, their value will be zero if $t_{i,s}$ matches in S. Otherwise, their value is the sum of the previously unmatched right completions $\hat{s} \cdot 0$ and $\hat{s} \cdot 1$ $(C_{i+1}[\hat{s} \cdot 0] + C_{i+1}[\hat{s} \cdot 1])$. This way, $C_1[s]$ will contain information about all the possible unmatched right completions.

Phase II: Generating strings unmatched by S. In this phase, the process will occur inversely as of the Phase I. The detectors will be generated by creating l-bit right completions of s with the information stored in $C_1[s]$. As $T = \sum_s C_1[s]$ is the size of the candidate detector set, each unique detector string is numbered from 1 to T. By drawing random numbers k, to find the k^{th} unmatched is to find successively the $\hat{s} \cdot b$ with $b \epsilon \{0,1\}$. This is achieved by jumping from intervals into intervals until finding $\left(\sum_{i=2}^{l-r+1} P_{i-1}\right) + C_{l-r+1}[s] = k$.

Example 1.1 Let $l = 4$ $\left(U = \{0: 2^l - 1\}\right)$, $r = 2$ and $S = \{s_1, s_2, s_3\}$, with $s_1 = \{0011\}$, $s_2 = \{0110\}$ and $s_3 = \{1100\}$. It is not difficult to verify that only one detector can be generated, namely $\{1001\}$. This implies that all bit strings that match the templates $\{10 **, * 00 *, ** 01\}$ are detectable as nonself (N'), which in this case corresponds to 8 strings. Inversely, all bit strings from the set $U \setminus \{S \cup N'\} = H = \{0010,0100,0111,1110,1111\}$ are not detectable, and considered as holes. If an additional string such as $s_4 = \{0001\}$ was also part of the self set S, no detectors can be generated and so, $D = \emptyset$ and no nonself strings are possible to detect.

A pseudo-code for the linear time algorithm is presented as follows.

Algorithm 1.2: Linear Time Algorithm.

input : $\boldsymbol{l}, \boldsymbol{r}, \boldsymbol{t} \in \boldsymbol{N}$, where $1 \leq \boldsymbol{r} \leq \boldsymbol{l}, \boldsymbol{t} = \boldsymbol{U}(\boldsymbol{r})$ and $\boldsymbol{t} \in [0, 2^r - 1]$
$\boldsymbol{s} \in \boldsymbol{S}$ and $\{\hat{\boldsymbol{s}}, \boldsymbol{S}\} \subset \boldsymbol{U}$

output : detector set $\boldsymbol{R}$

Phase I: Solving the counting recurrence

```
1  begin
2    for 0 ≤ s ≤ 2^r − 1
3          C_{l−r+1}[s] = { 0, if t_{l−r+1,s} is matched in S
                          { 1, otherwise
4    endfor
5    for i = (l − r), … , 1
6        for 0 ≤ s ≤ 2^r − 1
7            if t_{i,s} matches with any bit string of S then
8                    C_i[s] = 0
9            else
10                   C_i[s] = C_{i+1}[ŝ · 0]+C_{i+1}[ŝ · 1]
11           endif
12       endfor
13   endfor
14 end
```

Phase II: Generating strings unmatched by S

```
1  begin
2    The size of the total candidate detectors C: T = Σ_s C_1[s], where C_1[s] denotes the
     number of unmatched l-bit strings starting with the r-bit binary string s
3    for n = 1, … , N_R detectors
4      draw randomly k ∈ {1,2, … , T}
5      First interval ]P_1, Q_1] is calculated by finding s_1 such that
           P_1 = Σ_{s<s_1} C_1[s] < k ≤ Σ_{s=s_1} C_1[s] = Q_1
6      Determine the k^th unmatched string by evaluating
7      for i = 2, … , (l − r + 1)
8        if k ≤ (P_{i−1} + C_i[ŝ_{i−1} · 0]) then
9            P_i = P_{i−1}
10           b_i = 0
11       else
12           P_i = (P_{i−1} + C_i[ŝ_{i−1} · 0])
13           b_i = 1
14       endif
15     endfor
16   endfor
17 end
```

The time and space complexities depend largely on the construction of the matrix C. In Phase I, the time is consumed essentially in filling those entries that match self strings ($\mathcal{O}((l-r)\cdot N_S)$) and filling the rest of the matrix. In Phase II, it takes time proportional to r ($\mathcal{O}(r)$) to find the first r bits of the detectors, and ($\mathcal{O}(l-r)$) to complete the detector, for each of the N_R detectors:

time: $\mathcal{O}\big((l-r)\cdot N_S\big)+\mathcal{O}\big((l-r)\cdot 2^r\big)+\mathcal{O}(l\cdot N_R)$

space: $\mathcal{O}((l-r)^2\cdot 2^r)$

Therefore, the linear time algorithm runs in time linear with the size of the self and detector sets (N_S & N_R). However, given that it has a high dependency in l and r, using long strings and matching thresholds represents a problem both for time and space. The *linear time* algorithm is limited to the *r-contiguous bits* matching rule.

Generally, only a fraction of the candidate detectors is included in the detector set R, mostly because of overlapping. The *greedy* algorithm solves this problem by choosing first detectors far apart, matching unmatched nonself strings as possible [2, 3, 5].

1.4 Analysis

In order to analyse the features of the general mechanism of negative selection proposed by Forrest and collaborators, experimental results were obtained by numerical simulations. These simulations were performed using the linear time algorithm for detector generation and an additional "exhaustive" matching algorithm for counting holes.

The exact expression for the failure probability P_f is:

$$P_f = (N_N - N_D)/N_N \tag{1.17}$$

However, given that holes are undetectable, a lower bound for P_f (for each r) is:

$$P_f \geq N_H/N_N \tag{1.18}$$

Results from the simulations are presented in tables 1.2 and 1.3.

Table 1.2 shows how the probability of failure and the percentage of holes evolve when the parameters l and r are changed for a same self set. Results from self sets with a different number of elements (N_S=100 and N_S=150) are presented. The self set with N_S=150 was expanded from the N_S=100 set, by adding 50 extra strings. Strings were drawn from random numbers between 0 and 2^9-1. These same self sets will be used in chapter 4 to test cellular frustration algorithms. In this way the performance of the different approaches on the same conditions can be compared.

In Table 1.3, results obtained from self sets randomly generated from strings with 12 bits are presented. In order to analyse and eliminate the possible statistical variability from the previous example, 25 self set were tested and averages and standard deviations are presented. In these results the

dimension of the space was maintained relatively small ($l \leq 12$) because we tested results considering the whole set of detectors that could be defined, and also we tested all possible strings in the universe.

Table 1.2 Experimental results for different matching thresholds r, for two related self sets.
Strings of the self sets are previously defined within the range [0:2^9-1].

l	r	N_S	P_f	*holes*	N_R	N_S	P_f	*Holes*	N_R
9	9	100	0.000	0	2^9-100	150	0.000	0	2^9-150
	8		0.107	44	272		0.191	69	175
	7		0.340	140	89		0.674	244	24
	6		0.864	356	6		1.000	2^9-150	0
	5		1.000	2^9-100	0		1.000	2^9-150	0
10	10	100	0.000	0	2^{10}-100	150	0.000	0	2^{10}-150
	9		0.030	28	752		0.053	46	620
	8		0.083	77	459		0.184	161	255
	7		0.285	263	147		0.674	589	32
	6		0.827	764	12		1.000	2^{10}-150	0
	5		1.000	2^{10}-100	0		1.000	2^{10}-150	0
11	11	100	0.000	0	2^{11}-100	150	0.000	0	2^{11}-150
	10		0.014	28	1776		0.024	46	1644
	9		0.021	41	1408		0.033	63	1102
	8		0.071	138	856		0.130	247	447
	7		0.253	492	281		0.614	1166	56
	6		0.737	1436	24		1.000	2^{11}-150	0
12	12	100	0.000	0	2^{12}-100	150	0.000	0	2^{12}-150
	11		0.007	28	3824		0.012	46	3692
	10		0.010	41	3456		0.016	63	3150
	9		0.023	90	2740		0.036	141	2108
	8		0.063	253	1690		0.115	453	877
	7		0.162	647	558		0.476	1878	112
	6		0.684	2732	48		1.000	2^{12}-150	0

Table 1.3 Experimental results for different matching thresholds r, for 25 random self sets of size $N_S = 100$.
Strings of the self sets are within the range [0:2^{12}-1]. Standard deviation values are in parentheses.

l	r	N_S	P_f	*holes*	N_R
12	12	100	0.000	0	2^{12}-100
	11		0.002 (0.001)	6.8 (2.9)	3802.8 (2.9)
	10		0.006 (0.002)	23.5 (7.0)	3360.6 (15.8)
	9		0.020 (0.004)	78.0 (16.6)	2506.7 (41.5)
	8		0.064 (0.010)	254.0 (38.8)	1236.6 (52.4)
	7		0.281 (0.039)	1121.8 (156.3)	252.6 (57.1)
	6		0.849 (0.072)	3391.2 (288.0)	6.9 (6.0)

r	P_f	*Holes*	N_R
$\approx$ 6.6	$\approx$ 0.460	$\approx$ 1840	$\approx$ 100

The previous results are very interesting. First it is clear from Table 1.2 that increasing the size of the string space (increasing l), requires generating more detectors to cover the whole nonself space. However this increases the number of detection errors, i.e. the number of *holes*. In the examples presented in Table 1.2, the self set was confined to 2^9 different strings. As a result, even though the number of holes increases with l, the probability of failure decreases (in percentage). These observations are summarized in Figure 1.6 a) & b).

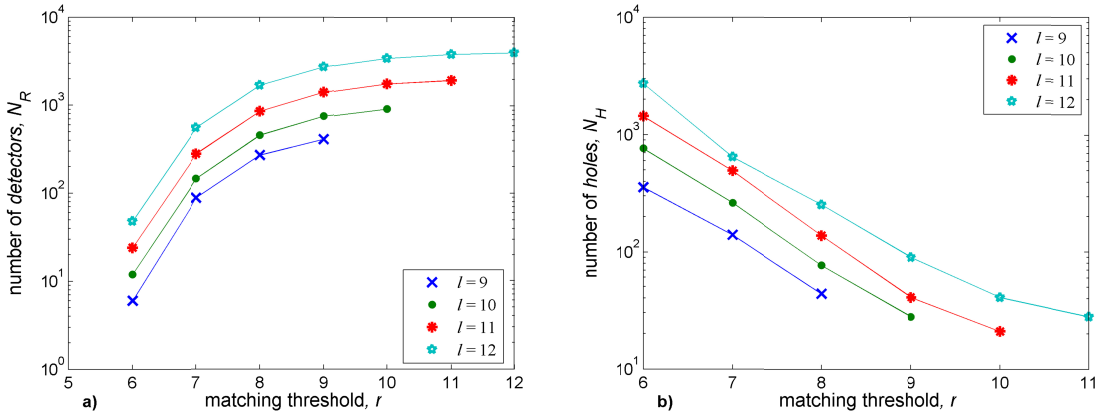

Fig. 1.6 Results from table 1.2 considering the first self set ($N_S = 100,\ S \in [0, 2^9 - 1]$) for different string space U defined by the string length l. Plots for: a) size of the complete detector set; b) number of holes created.

Comparing the results for the two self sets (table 1.2), it can be concluded that, for the same (l, r), the larger the self set, the larger the number of holes and the smaller the number of detectors required. This can be explained by the fact that when the number of self strings increases, the larger will be the number of discarded detectors during the censoring process. With a smaller number of available detectors, a larger number of holes are left, and hence P_f increases.

Another analysis can be performed by comparing the results obtained using N_S=100 self strings with $l = 12$ drawn from random numbers between 0 and $2^{12} - 1$ (Table 1.3) or from random numbers between 0 and $2^9 - 1$ (Table 1.2). Figure 1.7 b) shows a transition occurring when r is varied. For larger r higher detection performances can be obtained with the more disperse set of self strings. On the contrary, better results are obtained in the more confined self set, when r is small. However the detection performances are poorer in this case. Consequently, we can conclude that if high detection performances are required, then clustering of data could be detrimental. This could be important in practical applications, because general data sets are very often composed by clustered points dispersed in a sparse space.

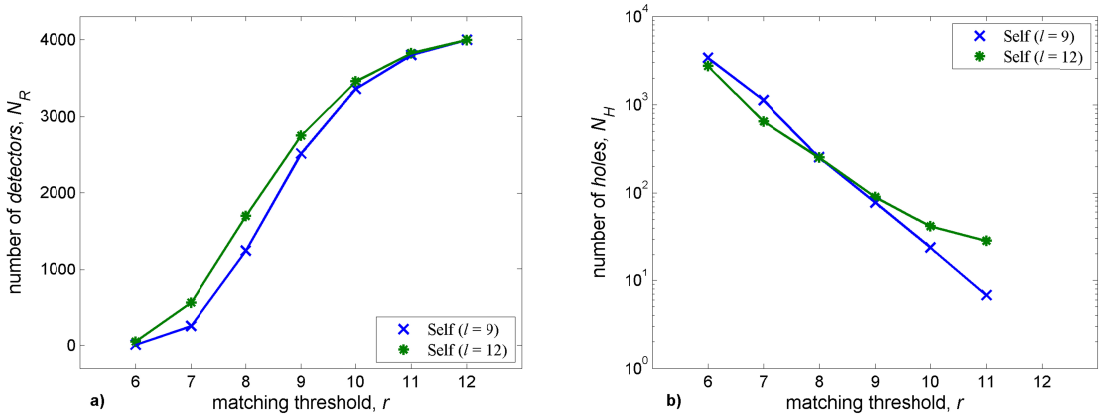

Fig. 1.7 Results from table 1.2 ($S \in [0, 2^9 - 1]$) and table 1.3 ($S \in [0, 2^{12} - 1]$), considering the self sets with $N_S = 100$ for the string space U defined by $l = 12$. Plots for: a) size of the complete detector set; b) number of holes.

In Figure 1.8 it is shown how the number of holes decreases when more specific detectors are used (r large), and how simultaneously the number of detectors that need to be considered to cover the whole nonself space grows. In that case, the probability of failure decreases with r (Figure 1.8b)).

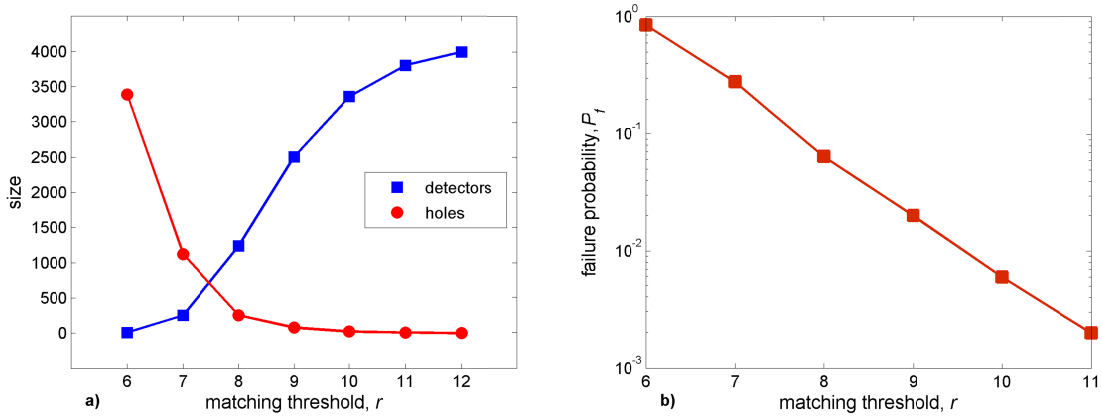

Fig. 1.8 Results from table 1.3 considering the mean of 25 random self sets ($N_S = 100$, $S \in [0, 2^{12} - 1]$) for the string space U defined by the string length $l = 12$. Plots for: a) size of the complete detector set and the number of holes created. b) lower bound of the failure probability.

For a comparison with results obtained in chapter 4 with a Cellular Frustration algorithm, we estimated the number of holes when the number of detectors is the same as the number of self strings ($N_S = N_R = 100$). I applied a smoothing spline to the values of the number of detectors on table 1.3, and found that in this case r is close to 6.6. To estimate the number of holes in this case, I applied an exponential fit and found that this number would be $N_H \sim 1840$ (table 1.3, right).

Another important analysis concerns how the probability of failure changes when the space dimension increases. This is particularly important for practical applications because the number of self strings can easily be much smaller than the number of possible strings: $N_S \ll 2^l$. Using similar fitting methods as in the previous case, I estimated from table 1.2, the number of detectors required to

achieve P_f=0.01, for each value of l from 9 to 12. The results are presented in Figure 1.9 (blue dots). It is clear from this plot that the number of detectors increases exponentially with l. For instance, it is possible to extrapolate that for l=32, about $2.44 \cdot 10^9$ detectors (N_R) would be necessary to achieve P_f=0.01. This shows that this type of algorithms become unpractical for space dimensions of this order.

A final remark should be made here. The later analysis was done using the values obtained for the confined self set in Table 1.2. Hence, we should expect that the former estimates could be an overestimation of the actual number of detectors required. However, the exponential dependency is maintained and consequently the qualitative nature of the conclusion remains valid.

Another interesting remark is the observation that in the previous l=32 example, it would be necessary that on average each detector would detect only 2 different nonself strings ($2^{32}/N_R \approx 1.76$). This shows that to achieve failure probabilities of the order of 1%, the number of detectors is indeed too large.

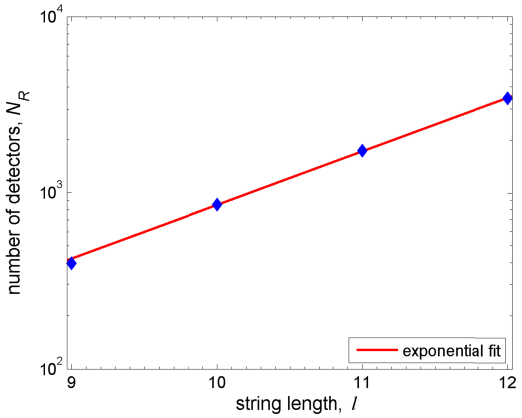

Fig. 1.9 Increase in the size of the complete detector set for a failure probability about 1%. Results calculated from the values of table 1.2 considering the first self set ($N_S = 100,\ S \in [0, 2^9 - 1]$) for different string space U defined by the string length l.

1.5 Final Remarks

The NSAs developed by Forrest and collaborators have an acceptable performance for the case in which the self size is small compared to the dimension of the nonself space, and for a low value of the matching threshold r compared to the strings length l. Even though, in the best case scenario, the number of detectors is at least 3 to 4 times larger than the size of the self set (as reported in [2, 3] using the greedy algorithm and for a $P_f \approx 0.1$). Smaller values of P_f, could also be achieved. However, the cost of such small failure probabilities is an unreasonable growth of the number of detectors. In this case, the size of the detector set becomes close to the size of the nonself space. In these algorithms, this leads to including many overlapping detectors. Another weakness of the NSAs is that the size of

the complete detector set increases exponentially with the size of the string space U, parameterized by the string length l - fig. 1.9.

The *exhaustive search* detector generation algorithm is very time demanding given that the censoring process is completely random. The *linear time* and the *greedy* algorithms, improve the detector generation phase but they are still slow because all the possible templates with $(l - r + 1)$ positions have to be matched.

The complete negative selection algorithm is still not efficient for two other reasons. First, in NSAs holes can only be counted *a posteriori* and this has a high computational cost. Secondly, the monitoring phase is not efficient for large detector set's size. The monitoring requires checking all the detectors one by one which is very time consuming for reasonable detector sets. It should be noticed that in most cases monitored strings do not correspond to intrusions. Consequently, the typical case requires testing all detectors, which can be burdensome.

The matching rules provide a framework for easily matching strings with reduced matching requirements. Regarding the r-contiguous matching rule (*rcb*), for a fixed string space U, there is a trade-off between the number of detectors and the failure probability P_f: for a higher matching threshold r more detectors are necessary to cover the same nonself space N; for a lower r, less detectors are necessary but more holes are created (higher P_f). This trade-off corresponds to tuning between the specificity and generality of the detector coverage. There is an "optimal" choice of parameters that produces an acceptable number of holes (low P_f) with a small detector set.

There are also some questions raised about the immunological relevance of these algorithms. Nevertheless, the main purpose of Forrest and collaborators is not to raise hypothesis on immunological foundations. According to the theory T cells would not bind self cells after the thymus. This, however, is not observed, and indeed a wide range of autoimmunity is observed.

Furthermore, an important problem in these approaches is that there is not homeostatic control of self. In NSAs holes are structurally maintained around self (fig. 1.4). Consequently, these structural faults could be a persistent problem that could be used by pathogens using molecular mimicry strategies. This means that if an intruder uses only self patterns or also patterns in holes, then it could proliferate without control. To minor this problem it would have to be considered an extra mechanism for monitoring or counting holes.

Another issue concerns the problem of choosing the best value of r. It could be thought that the best strategy would be to consider detectors with different values of r. This however raises the problem preferentially select detectors with low r, that cover larger regions of space. Otherwise the system would end by having plenty of detectors that cover regions already covered by lower r detectors. It would be necessary to have a mechanism of evaluating detectors overlapping, such as in the *greedy* algorithm. This, however, adds another level of computational complexity in the algorithm.

Negative selection algorithms can be classified between signature scanners and file-authentication programs [1]. There are some authors that tested specifically the exhaustive search algorithm for real applications of network intrusion detection systems. These authors used high values for l, r, and also larger coding alphabets, reporting that for some cases it would take years to generate a certain number of detectors [8, 9]. These authors also encourage the use of other algorithms or together with the NSA, such as, e.g., the use of positive selection, clonal selection, other matching rules, etc.

There are many other variations of the NSAs [5] namely, introducing mutation in detector generation, using crossover methods to obtain the holes, using different types of representations (e.g., "negative databases"), making use of other immunological structures, and so on. The field is prone to variations, but qualitatively different results should not be expected.

Chapter 2

Cellular Frustration

The conceptual framework of Cellular Frustration was first developed by *de Abreu et al.* [10] as a new phenomenological understanding of the adaptive immune system. This mechanism presents an explanation for self/nonself discrimination taking into account "cellular decisions". Frustration in T cell interactions with self APCs provides a basis for the regulation of the immune responses.

In this chapter, the Cellular Frustration mechanism is presented extending the immunological concepts introduced in the previous chapter. This mechanism is compared with approaches on the same problem by Forrest and collaborators. A model for nonself detection is described by using the concepts of Cellular Frustration. An algorithm based on this model is developed in chapter 4.

2.1 Introduction

The problem of self/nonself discrimination has been tackled by many different theories. The best know approaches include negative selection, idiotypic networks, clonal selection or even the danger theory [4-6, 11, 12]. The common feature that some of these approaches share is that affinity is related to the complementarity of structures in ligand-receptor binding. They also share the knowledge that reactions are essentially instantaneous. These features are "appealing" on a computational point of view for Artificial Immune System (AIS) algorithms, given that they are easy to implement.

However, immune reactions require time: either during recognition processes, or for effector functions to take place [11]. The complex immunological synapse that is established when an APC forms a conjugate with a T cell, is a demonstration of those time lasting processes. The immunological synapse involves complex signalling and intermediate conformational processes preceding T cell activation [13]. The kinetic proofreading models are good examples of multi-step

signalling pathways required for very specific processes, like DNA replication [14] or T cell activation [7, 13, 15].

T cell activation requires that the immunological synapse is preserved for some long characteristic time. Only after T cell activation can effector functions take place. So, in order not to kill self cells, there should be a process to avoid T cell activation when T cells are bound to APCs carrying self antigens. This is necessary to maintain self in absolute tolerance. Likewise, APCs carrying nonself antigens should establish long lasting interactions with T cells, leading to a pathogenic recognition process and triggering an immune response against the pathogen. This would require extreme reactivity only against nonself cells: to increase the probability of long-lasting interactions or to increase reaction speed.

Cellular Frustration accomplishes these two apparently incompatible tasks: to react very specifically with nonself, while maintaining non-reactivity (tolerance) with self. These features are achieved by cellular frustrated systems exhibiting decision-like interactions.

Consider the following example with 3 different cell types (A, B, C). Suppose that each type has different affinities and that they can be described by an Interaction List (IList), such as the one in fig. 2.1a). ILists can be regarded as lists of "preferred interactions" with the highest preferences at the first positions (for each cell type). Cells display different affinities for each different cell type, i.e., have more probability to react with some cells than with other cell types. If a cell is interacting with one other, their interaction can be interrupted if a different cell appears. In this sense, cells become frustrated if their immunological synapses are consequently interrupted to form new conjugates with another cell.

Within the Cellular Frustration framework it is assumed that cells perform decisions. These "decisions" are the output of optimization processes based on the avidity of cell contacts. Cell contacts are mediated by ligand-receptor interactions. This phenomenon has indeed been observed by *Depoil et al.* [16]. These experiments show that an immunological synapse can be changed towards a new APC, if this one provides a stronger stimulus.

The IList in figure 2.1a) shows how affinities are sensed by cells in a frustrated set. If cell A is prompted with one cell B and one cell C, cell A decides to interact with cell B. In the same way, cell B prefers to interact with cell C than with cell A, and so on. In this system no stable arrangement is achieved, given that affinities are not reciprocally matched. This leads to a frustrated interaction cycle where no conjugate of two cells remains bound for too long. This cycle is illustrated in fig. 2.1b). A cellular frustrated system typically exhibits a majority of short-lived interactions, because no stable configuration exists. The system of 3 cell types, with the configuration of the IList from fig. 2.1a), is maximally frustrated.

a)

A	B	C
B	C	A
C	A	B

b)

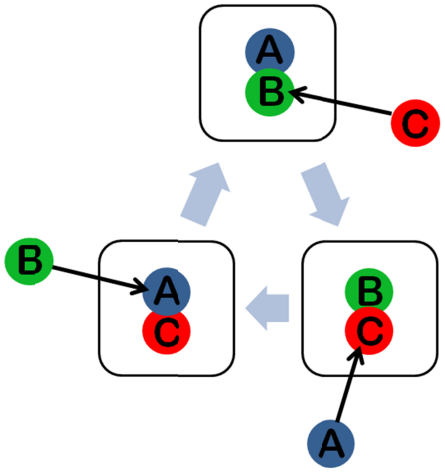

Fig 2.1 Cellular Frustration in ABC system. a) Interaction list for ABC system. b) ABC system exhibiting cellular frustration based on the IList of a). Conjugates are constantly destabilized by single cells, making no stable arrangement possible (frustration cycle).

Consider now what would happen if another cell type D is introduced in the previous ABC system. Assume that cell D is similar to cell type A in that it displays the same IList, i.e., it has the same receptors. However, it presents different ligands and so, the other cell types can distinguish them when confronted with the decision between interacting with A or D cells. This is featured in the IList of fig. 2.2a). In this system, cell A will remain maximally frustrated (as in ABC system), given that its IList remains oppositely matched with cells B and C. However, reactivity is increased towards cell D, by other cells. This happens because cells B and C do not frustrate so much cell D. Cell D assumes intermediate positions in their ILists. Therefore, BD conjugates will have longer lifetimes, given that B has D in the first position of its IList (fig. 2.2a)).

The system with 4 cell types (ABCD) can provide the basis for a model on how the adaptive immune system is activated in the spleen and lymph nodes [11]. The three cell types A, B and C, correspond to self APCs, T cells and regulatory T cells (T-reg), respectively. In this sense, the cell type D is similar to A but, corresponds to pathogenic APCs, i.e., APCs presenting nonself peptides. A longer conjugate time of cells B with cells D triggers T cell activation (fig. 2.2b)). The rest of the system remains in a tolerant state avoiding auto-immunity. Figure 2.2b) illustrates some of the possible interaction results from single cell destabilizations within the ABCD system.

A multi-step system based on the ABCD dynamics for T cell activation has also shown to possess similarities with the T cell kinetic proofreading by *McKeithan* [15]. This kind of system has also shown other useful characteristics namely, tunable specificity controlled by the chemical species' concentrations [19]. However, this topic is not the subject of the present work.

a)

A	D	B	C
B	B	C	A
C	C	D	D
-	-	A	B

b)

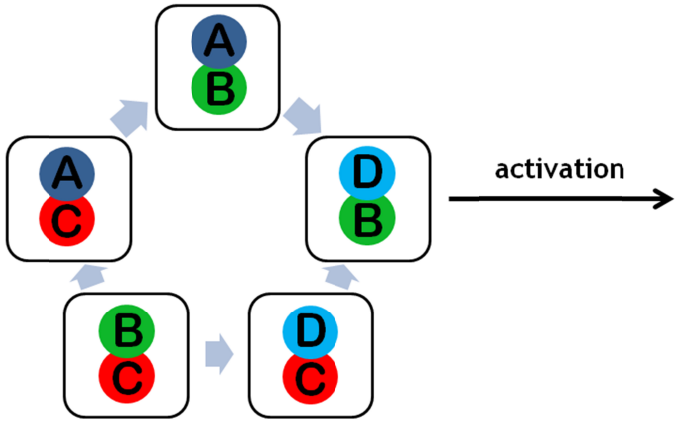

Fig 2.2 Cellular Frustration in ABCD system. a) Interaction list for ABCD system. b) Schematics of all cell conjugates of ABCD system and two possible destabilization pathways using single cells, leading to conjugate BD (containing pathogen B). Conjugate AB can be destabilized by a D single cell. Therefore, conjugate BD is more stable than AB, producing longer lifetimes which lead to T cell activation. The interactions are based on the IList in a).

The Cellular Frustration mechanism was proposed as an explanation to these phenomena, breaking with all previous theories proposed on the immunological self/nonself discrimination. Actually, frustration is a natural occurring phenomenon in many different systems. Examples could range from neural networks to spin glasses [17]. Even the very idea of Cellular Frustration was inspired by ecological mating dynamics [18].

In the Cellular Frustration mechanism, the notion of self and nonself are emergent properties from the system dynamics. In this framework, self is defined as the set of cells that keep short-lived interactions (frustrated dynamics). This dynamics is dependent on the ligands and receptors present in the system [10, 11]. In the same way, cells that keep long-lived interactions are considered nonself (non-frustrated dynamics). One of the major differences that clearly distinguishes this mechanism from others, is that all elements are treated as similar, having the same rules and constants applied.

The Cellular Frustration framework relies on some assumptions that extend the immunological concepts introduced in chapter 1, namely:

Pathogen mimicry: APCs should all be similar, independently to whether they are displaying self or nonself antigens.

Cross-reactivity: each immune cell can potentially interact and react with a large variety of cells. It is also assumed that immune cells have high avidity. This means that, cells prefer to be in a conjugate state than to be alone. Furthermore, cell interactions can be conflicting because cells' decisions are not necessarily matched by the other cells.

Cellular decisions: cells can only interact with one cell at a time. It is assumed that cells perform decision-like interactions, directing the immunological synapse towards the cell from which they sense higher affinity.

Effector functions: immune responses take place after T cell activation. T cell activation can only be triggered if cells interact for a sufficiently long time (a characteristic time).

At this point, it is possible to draw some differences with the approaches by Forrest and collaborators and their immunological motivation.

Table 2.1 Comparison between important features of the two immunological frameworks: negative selection and cellular frustration

	Negative Selection (Forrest et al.)	***Cellular Frustration (Abreu et al.)***
Tolerance/ Auto-immunity	guaranteed when detectors match no self cell	guaranteed if all interactions are short-lived
Cross-reactivity	each detector has an affinity domain (limited)	each detector can in principle interact with any other cell (global)
Self	everything that is presented in the maturation stage (before repertoire selection)	
Nonself Recognition	a string (peptide) that lies within a detector affinity domain (detector-dependent property)	long-lived conjugate (emergent property)

2.2 N System and Maximal Frustration

Returning to the ABC system, explained before, this kind of system can be generalized for N cell types instead of the initial 3 cell types. In a similar way, we now consider structurally equal systems by having only one cell type and N different cell subtypes (denoted as N clusters).

The following considerations assume only one cell per subtype each having different ILists. Thus, assuming that all cells play equivalent roles, their ILists for a maximally frustrated system are obtained by the following relationship:

$$L_{[i+1]}(j) = L_i(j+1), \qquad j = 1, \dots, N-1 \tag{2.1}$$

with,

$$L_i(N) = i \tag{2.2}$$

Here $L_i(j)$ corresponds to the cell subtype ranked in the jth position in the IList of cell subtype i; and $[i]$ represents an integer i modulus N. Systems with this particular arrangement of ILists are

called circular frustrated systems. These systems can be represented as illustrated in figure 2.3, the ILists are sequential and interconnected. ILists from consecutive cell subtypes are obtained with minimal shifts assured by the continuity of cell elements.

This kind of system was constructed to be maximally frustrated because each cell's IList is oppositely matched by the ILists of the other cells. For instance, if cell subtype i puts cell subtype j in the first position of its IList, then cell subtype j has i in the last position of its IList. Expression 2.3 holds the generalization that if cell subtype i has cell subtype j ranked in the kth positions of its IList, then j ranks cell subtype i in $(n-k)$th position of its IList:

$$\begin{gathered} i = L_u(N-j), \qquad u = L_i(j) \\ i = 1, \dots, N, \;\; j = 1, \dots, N-1, \;\; L_i(N) = i \end{gathered} \tag{2.3}$$

As a result of these ILists, no conjugate is stable: there is always one cell within a conjugate that can be destabilized by one other cell of higher affinity (rank). These systems maximize frustration.

The maximum lifetimes displayed by maximally frustrated dynamics are the shortest lifetimes possible, for each given system. However, the complexity of interactions in these systems originates a distribution of short lifetimes instead of a single lifetime [11].

Any new cell introduced in the system with an IList not following the system's IList, decreases the system's frustration. This does not imply that all cells of the system become less frustrated. This can be straightforwardly understood by considering a simple example: this could be the case if we a new cell that is so little frustrated that it would be bound to a single cell in the system always. In that case, we would have a system composed of a permanent pair and the remaining cells whose ILs follow fig. 2.3a). These other cells would therefore remain highly frustrated and maintain short-lived interactions.

To further understand the concept that is behind nonself discrimination in this kind of systems, the concept of pathogen is clarified. Assume that the self system organizes itself according to some rule. This rule can be assigned by an education process or by a hierarchical system. Each cell can only see its own IList and not the system globally. A pathogen is an invader cell that does not belong to the system and therefore is nonself. The pathogen will try to find one of the system's IList to be mistakenly identified as self.

This system can be represented by the following situation. Imagine some vault system that has different passwords assigned for every authorized user (self). Every user only knows its own password and not the other users' passwords or how they were generated. An unauthorized user (nonself) that wants to access the vault will have to find one password that matches the security system as an authorized user. If the unauthorized user has no previous information about the rules of the password assignment, he has to generate random passwords.

Consider the following three sequential cases, assuming that the initial system consists in the $N = 6$ cell subtypes as shown in figure 2.3.

Case 1. In the first case, consider that all N cell subtypes are present.

Case 2. In second case, consider that cell #3 is removed ($N = 5$).

Case 3. In the latter case, consider that cell #3 is removed and an invader cell (pathogenic) is introduced in this system.

a)

1	2	3	4	5	6
2	3	4	5	6	1
3	4	5	6	1	2
4	5	6	1	2	3
5	6	1	2	3	4
6	1	2	3	4	5
1	2	3	4	5	6

b)

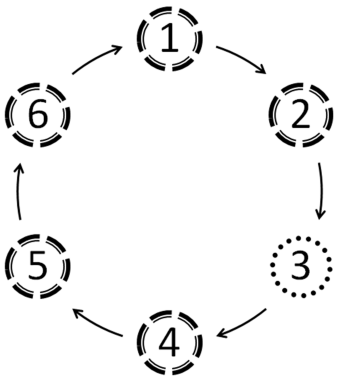

Fig 2.3 Circular frustrated system. a) Interaction list for a generalized maximally frustrated system ($N = 6$). b) Circular frustrated system representation, illustrating the IList in a). For instance, if cell #3 is removed, the new system remains circular and maximally frustrated.

The first case, holds a circular frustrated system (fig. 2.3b)) built using the IList in figure 2.3a). This system is maximally frustrated and therefore shows the shortest conjugation lifetimes. The dynamics of this system is the ideal self dynamics.

In the second case, if the IList is unchanged, the new system is still maximally frustrated. This is a feature of the circular frustrated systems.

The last case is to consider that self is confined to the 5 cell subtypes left (#3 removed, $N_S = 5$) and then a pathogenic cell (nonself) is introduced. Assuming that the self ILists remain unchanged, two situations can occur. If the nonself IList matches the system's IList (i.e., previous cell #3, fig. 2.3a)), the system will remain maximally frustrated. In this situation, the pathogen is matched as self. The second possible situation is that the pathogen's IList does not match the system's IList. In this situation, the interactions of the nonself cell will not be as frustrated as in the first situation. Consequently, longer lifetimes appear within the conjugates involving the nonself cell. This situation provides the basis for the nonself discrimination that leads to nonself detection, in this kind of systems. Nonself will be detected by the production of longer lifetimes. Assuming that there is only one nonself IList that matches the self system's IList, in order to maximize frustration, the probability of matching the system's IList is given by:

$$P = 1/N! \tag{2.4}$$

This probability is very small, decreasing largely with the number of cells involved. This reveals that virtually is almost impossible a pathogen not to be detected in a maximally frustrated system. Detection of pathogens is perfect in maximally frustrated systems, as shown by *de Abreu et al.* [11] both theoretically and numerically.

2.3 A Generalized model for nonself detection

A model for nonself detection is now outlined. This model is based on the PhD work under development by Patricia Mostardinha.

The motivation for this model is the same as addressed by Forrest and collaborators. The model allows having a set of agents to monitor a given data set (self). The function of the complex agents is to detect anomalous data (nonself). Nonself data can be positioned in the surroundings of self data, so that it is hard to discriminate both using classic negative selection methods.

This can be achieved by the fundamental functions of T cells and APCs in the immune system. Cellular frustration is reduced into two classes: T cells, working as recognition detectors; and APCs (self and nonself) working as information presenters. The situation is now the following.

A hypothetical situation is given to represent this system. Consider a vault system where a set of security agents monitor the vault users. Each user, authorized (self) or not (nonself), has a password. Furthermore, unauthorized users try to behave as an authorized user relatively to each security agent. Each security agent has a different complex rule that manages the authorized users' passwords. An unauthorized user (pathogen) will try to find passwords similar to authorized users' passwords. However, the unauthorized user has no previous information about the authorized users' passwords or how they were generated. Because of this fact, the unauthorized user will have an anomalous behaviour to one or more of the security agents and therefore will be easily detected.

The model is based on some of the considerations for nonself discrimination presented for the generalized N system (circular frustrated). T cells will have an associated IList where all APCs are ranked, and vice-versa. It is considered that cells do not interact with other cells from the same class. The idea is again to frustrate the self system so that, when a pathogen is introduced in the system, it will be detected given that it is not as frustrated.

The limit case for this system is the maximal frustration. If this system is maximally frustrated, it has the features of a double circular frustrated system, as illustrated in figure 2.4. This system has perfect detection for the same reason as explained for the circular frustrated systems. The ILists in figure 2.4 verify relationship (2.3), having in mind that each IList links to the elements of the other class.

For this model additional assumptions are also considered, which are listed as follows:

1. The number of detectors is equal to the number of self presenter cells ($N_R = N_T = N_S$).
2. APCs are kept with the IList corresponding to maximal frustration (fig. 2.4).
3. The dynamics of APCs are independent of the peptide self/nonself classification.
4. APCs and T cells, both make decisions and so, will have a ligand and a receptor assigned to each cell.
5. T cells' IList are assigned by an *education* process.

Assumption #1 states that the number of detectors N_R is a well-defined quantity, which is completely different to the assumptions of Forrest and collaborators. The size of the detector set is considered to be equal to the size of self N_S.

From assumption #2 and #3, it is established that APCs (self and nonself) have a previously defined IList (genetically encoded) that promotes the organization onto a maximally frustrated dynamics (similar receptors).

Assumption #4 considers that APCs and T cells read and present information, by having on its surface both a receptor and a ligand. Receptors read information displayed by ligands.

By assumption #5, T cells' IList are initially randomly generated. However, T cells must go through a selection process to maximize frustration with self. Only T cells with ILists displaying the shortest lifetimes should be left in the final repertoire. The T cell maturation process (*negative selection*) will be designated by *education*.

Considering the size of self (N_S , number of self APCs), the probability that the complete IList of T cells ranks randomly the pathogen with maximal frustration (circular frustrated), is:

$$P_{\text{max\_pat}} = \left(\frac{1}{(N_S + 1)!}\right)^{N_S} \tag{2.5}$$

Expression (2.5) shows that it is virtually impossible for all T cells to randomly generate the complete IList that reaches the maximal frustration. This probability is completely dependent on the size of self and a small increase in this variable has a high decrease in the probability value.

For instance for a small size of self, $N_S = 5$, $P_{\text{max\_pat}} \sim 5 \cdot 10^{-15}$. This result is already infeasible by random chance. For larger N_S the maximal frustration regime is increasingly hard for the pathogen to achieve. This shows that nonself detection is increasingly easier with an increasing size of self.

The question relies in whether it is possible or not, to achieve by an optimization process, a regime outside maximal frustration with self, in which the variations are not crucial to change the accuracy in nonself detection.

APCs					
1	2	3	4	5	6
1	2	3	4	5	6
2	3	4	5	6	1
3	4	5	6	1	2
4	5	6	1	2	3
5	6	1	2	3	4
6	1	2	3	4	5

Ts					
1	2	3	4	5	6
2	3	4	5	6	1
3	4	5	6	1	2
4	5	6	1	2	3
5	6	1	2	3	4
6	1	2	3	4	5
1	2	3	4	5	6

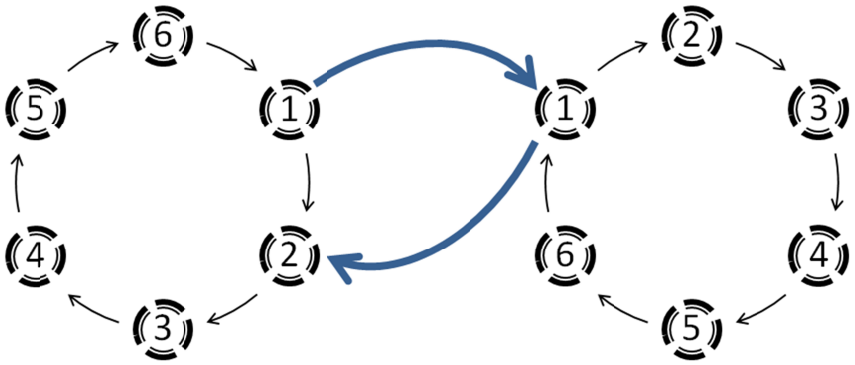

Fig 2.4 Generalized model for nonself detection consisting of two cell classes: APCs and T cells. ILists for maximal frustration and its representation (double circular frustrated system). APC from cluster #1 ranks T cell from cluster #1 in the first place. However, in order to maximize frustration, T cell from cluster #1 ranks in the first place APC from cluster #2, putting APC from cluster #1 in the last place of the rank.

The Cellular Frustration algorithm that is developed in chapter 4, uses this model working outside the maximal frustration limit. An *education* process precedes a *monitoring* phase. In *education*, T cells initially random, will evolve their IList to approximate maximal frustration with self. The aim is to keep self as frustrated as possible. In maximal frustration, T cells rank APC clusters (peptide types) as in figure 2.5. By *education*, T cells are selected to rank APC clusters approximately with the relative positions as ranked in figure 2.5, as to maximize frustration. This is because pathogens can be ranked in the middle positions among the ILists.

In the monitoring phase, pathogenic cells are introduced in order to test nonself detection. A pathogen will be an APC from a different cluster of the self clusters existent (different peptide). When arbitrary clusters are generated, then a pathogenic cell can be generated featuring a cluster in the middle. Pathogens will be detected by exhibiting a higher number of longer lifetimes when conjugated with T cells (in average). Self APCs' lifetimes tend to be lower because they are more highly frustrated.

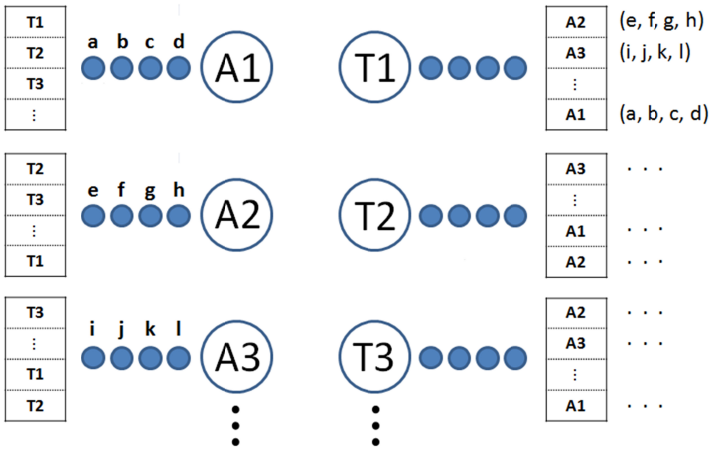

Fig 2.5 Features of the Cellular Frustration model. The two classes of cells, APCs and T cells, showing ILists for maximal frustration. T cells rank APC clusters with numbers 1, 2, 3, etc. In the event of having cell sub-clusters (lower case letters, a-l), they should be ranked inside the corresponding cluster rank. Sub-clusters can be ranked in the same rank position or in randomly distributed rank positions preserving the clusters' relative positions.

2.4 Final Remarks

Cellular Frustration mechanism provides a new approach to understand how the immune system works, specifically on how self/nonself discrimination can be achieved while maintaining absolute tolerance with self. Frustration in cellular decisions is considered in order to deal with apparently incompatible features: reacting strongly towards pathogen-derived peptides while not reacting at all to self peptides.

Unlike the immunological foundations taken into account by Forrest and collaborators, Cellular Frustration's nonself recognition is possible due to emergent dynamical properties. The system's own dynamics provides self and nonself to be defined as an output of a complex system of affinities. This approach states that nonself discrimination arises from a time consuming optimization process rather than from simple structural matching. It was demonstrated theoretically that with Cellular Frustration it is possible to achieve perfect nonself discrimination, using the maximal frustration case. Numerical confirmations of this result motivated the development of new algorithms for nonself detection, which is the aim of this thesis.

A model for nonself detection using cellular frustration was sketched. This model uses only two classes of cells: T cells and APCs. These classes are used by their ultimate function namely, T cells work as detectors, and APCs work as information presenters. A learning process (*education*) selects T cells to rank self cells as to maximize frustration. Because pathogenic cells were not present during the education process, they will subsequently occupy random positions in the T cells' ILists, becoming less frustrated when included in the system. Afterwards, in a *monitoring* phase pathogens

can be detected by analyzing the time behavior of all the system's conjugates. Pathogens will display a higher number of longer lifetime events, relatively to self conjugates.

The development of an algorithm using this model has numerous advantages over the majority of negative selection algorithms. The algorithmic method considered by Forrest and collaborators consists on patching all the nonself space with detectors.

The Cellular Frustration model approaches this problem in a completely different perspective and has also advantages. The Cellular Frustration mechanism can trigger detection events even when there is an abnormal growth of self cells [11]. This feature is very interesting both immunologically and numerically. This is a considerable advantage over negative selection algorithms, as these permanently discard self by not monitoring it. Consequently, if a pathogen exists that does not use nonself peptides then it could propagate freely.

The Cellular Frustration model minimizes the number of detectors. The number of detectors considered is of the size of the self set, which sets a major difference to Forrest and collaborators. In Cellular Frustration detectors are global, given that they rank all presenter cells.

It is also reported in *de Abreu et al.* [11] that the detection time required remains of about the same magnitude the number of detectors is increased. This contrasts with the approaches by Forrest and collaborators, in which the probability of matching decreases with a higher size of the detector set.

An algorithmic version of the model is developed in chapter 4, including numerical results and further comparisons with Forrest and collaborators.

Chapter 3

Diversity in Interaction Lists

The original Cellular Frustration formulation used Interaction Lists (ILs) to map all possible, very diverse interactions. In practice, ILs consist in ordering of integers, each integer being associated to a cell number. We want to formulate mathematical algorithms that could compress the information contained in these Interactions Lists in simple mathematical rules using bit string information in ligands and receptors. However, in using only a few sets of strings to code the information contained in ILs, considerable information tends to be lost. In particular, not all types of ILs can be generated, and hence only a fraction of its diversity is preserved. In this chapter we will study how different rules can be defined and how diverse are the different ILs they can generate.

3.1 Full Interaction Lists

Cellular Frustration considers the cross-reactivity among a large number of cells in the immune system. Interactions are based on different affinities that can be ranked in Interaction Lists (ILs). The original approach to study the Cellular Frustration mechanism uses ILists that consider all possible orderings of cells in the system. For a system with N cells, their total number is:

$$N_{ILs} = N! \tag{3.1}$$

Consider now the generalized Cellular Frustration model for nonself detection discussed in the previous chapter. Given that N_S is the size of self and, N_{NS} is the size of nonself (number of pathogens), the total number of possible T cell's ILists is:

$$N_{ILs} = (N_S + N_{NS})! \tag{3.2}$$

ILists that promote maximal frustration in the Cellular Frustration model for given self sets were presented in chapter 2 (figs. 2.4 & 2.5). However, these ILists only contemplate the self cells. When

pathogens are introduced they will be ranked by T cells among self cells' positions. Nevertheless, maximal frustration with self is kept if the relative positions of the self APCs in the initial ILists are preserved. The total number of ILists preserving the relative positions of the self APCs in the rank, that keep maximal frustration, are given by:

$$N_{IL^*} = \binom{N_S + N_{NS}}{N_S} \cdot (N_{NS}!) = \frac{(N_S + N_{NS})!}{N_S!\ N_{NS}!} \cdot N_{NS}! = \frac{(N_S + N_{NS})!}{N_S!} \tag{3.3}$$

The number of ILists N_{IL^*} is relative only to each T cell's IList individually. Considering the introduction of a single pathogen, the number of ILists is given by:

$$N_{IL^*} = \binom{N_S + 1}{N_S} = \frac{(N_S + 1)!}{N_S!} = N_S + 1 \tag{3.4}$$

In the previous chapter, it was shown that it is increasingly hard for the pathogens to be ranked as to be maximally frustrated. On the other hand, expressions (3.3) and (3.4) when compared to expression (3.2), show that it is increasingly hard to achieve ILists with the self maximally frustrated. The percentage of these ILists in the total number of ILists (3.2) decreases with the size of self and nonself.

There is also the case when the self and nonself are confined to a string space. In the case that binary coding is used, the number of ILists depends on the number of bits. The total number of ILists possible to generate with p bits for a generalized 2^p cell system is:

$$N_{IL} = 2^p! \tag{3.5}$$

3.2 Univocal Binary Rules

The Cellular Frustration algorithm that is presented in the next chapter makes use of univocal binary rules to generate ILists. These binary rules are functions that modify the input strings. Thus, it is possible to order strings in different ways, producing different ILists.

The aim is to generate simple rules that use minimum information to generate the maximal diversity. This supposes that indeed these rules can generate considerable redundancy, and consequently many ILs can be, in principle, generated from different inputs. On the other hand, these ILs can encode, from the start, a smaller amount of information, for instance, because they use less bits than those required to consider all possible N numbers orderings.

Any data type can be represented as a sequence of bits in a computer memory. The use of binary representation is widespread due to its straightforward implementation. In this thesis, only binary coding was considered. Nevertheless, any type of data and codification can be used in the Cellular Frustration algorithm.

The importance of using univocal rules is to preserve the same number of elements with no repeated strings. The binary rules should preserve a minimum diversity in the generated ILists. A minimum number of different ILists has to be generated so that the pathogen can be less frustrated by average in that ILists. This means that, in order to obtain good detection, the pathogen must be ranked in the top positions, increasing the probability of stable conjugates with longer lifetimes.

Consider the matrix X, that consists of elements $x_i, \{x_1, \dots, x_N\} \in X$, and that is transformed onto matrix Y ($Y = f(X)$) which has elements y_i, $\{y_1, \dots, y_N\} \in Y$. The transformation rule applied to matrix X, is univocal if it verifies:

Condition 1. $$|X| = |Y| = N \tag{3.6}$$

Condition 2. $$X \neq Y \tag{3.7}$$

Condition 3. $$\forall i, \forall j, i \neq j : x_i \neq x_j \wedge y_i \neq y_j \tag{3.8}$$

Condition 1 states that the cardinality (size) of the initial matrix X must be equal to the matrix after the transformation rule is applied (Y). The same number of elements must be kept. Condition 2 represents that the matrix X is different from the matrix Y. The elements of both matrices can be the same, but positions must be different. Condition 3 states that if matrix X consists of unrepeated elements, matrix Y is also consisted of unrepeated elements.

In this sense, for the case of binary operations, a univocal binary operation guarantees that the output preserves the initial number of ones and zeros.

3.2.1 *TWR* Rule

The first rule operates using a simple set of operations on a given binary string X, to compute a score using information on two types of strings. Each operation has an associated string: two strings of s bits (called "toggle" and "reference") and a string of s positions (called "weights"). The latter string with s positions has integers between $0: s-1$, the bit significances of a s bits binary string.

1. **T**oggle, Tog;
2. **W**eights, W;
3. **R**eference, Ref.

The construction of ILists is based on the score value. The *TWR* rule implicitly orders all given strings X according to their different scores (fig. 3.1), given by:

$$score = (X \oplus Tog)_W - Ref \tag{3.9}$$

The sequence of this calculus is exemplified in figure 3.1. In this expression $X \oplus Tog$ operates the XOR ("exclusive or": $\oplus$) logical operation between bit strings X and Tog. The $()_W$ operation computes the string result of $X \oplus T$ by changing the binary string bit significances according to W. Each bit position is reassigned according to new positions ordered in string W. The third operation

applies the difference (subtraction) between the string result of the previous two operations and another one named Ref.

The univocal operations used provide unrepeated outputs by themselves when single functions. However, the combination of these functions produces some repeated ILists, which represents a loss of diversity. Results in terms of ILists diversity is shown in table 3.1.

Table 3.1 Number of ILists for the *TWR* rule.

s bits	2	3	4	5	6	≥7
total Ils ($2^s!$)	24	~$4.03\text{x}10^{4}$	~$2.09\text{x}10^{13}$	~$2.63\text{x}10^{35}$	~$1.27\text{x}10^{89}$	∞
total ILs with 3 rules ($2^s \cdot 2^s \cdot s!$)	32	$3.84\text{x}10^{2}$	~$6.14\text{x}10^{3}$	~$1.23\text{x}10^{5}$	~$2.95\text{x}10^{6}$	…
unrepeated ILs with 3 rules	16 (50%)	$1.92\text{x}10^{2}$ (50%)	~$3.07\text{x}10^{3}$ (50%)	~$6.14\text{x}10^{4}$ (50%)	~$1.47\text{x}10^{6}$ (50%)	…

For $s\ bits = 3$, the percentage of ILists generated with *TWR* rules relatively to the total number of ILists is only of 1 per cent. This percentage decreases astronomically with small increases in the number of s bits. The percentage of unrepeated ILists is 50 per cent of the total ILists possible to generate with *TWR* rule.

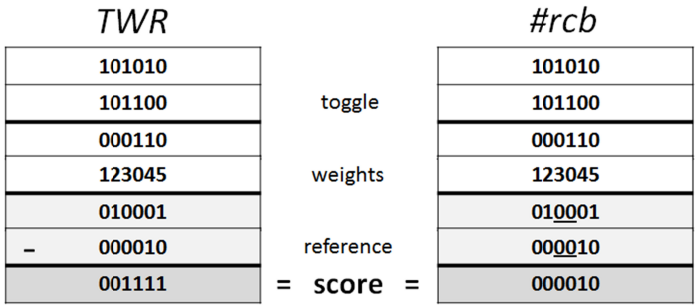

Fig 3.1 Example for the score calculus between *TWR* and *#rcb* rules, showing the difference between the two calculations.

3.2.2 *#rcb* Rule

The *#rcb* rule only differs from the *TWR* rule in the last operation used in the score calculus. This operation is inspired in the r-contiguous bits matching rule. The r-contiguous bits matching rule, as explained in chapter 1, considers that two sequences match if the first has a number of bits p that match at least r bits of the second ($p \geq r$). However, the operation here considered is not equal to the r-contiguous bits matching rule. Here, the operation considered is the distance measurement of similar of contiguous bits. The last operation counts the number of contiguous bits between the result from the previous two operations and the Ref string. For a binary string X, the score is a function of:

$$score = f_{\#rcb}((X \oplus Tog)_W\ ,\ Ref) \tag{3.10}$$

In this rule, only the first two operations are univocal. The third operation produces repeated elements because equal scores are possible. This introduces degeneracy among a certain number of elements. This results in a higher loss of diversity which is expressed in the results of table 3.2.

Table 3.2 Number of ILists for the *#nb* rule.

s bits	2	3	4	5	6	≥7
total ILs ($2^N!$)	24	~4.03x10^4	~2.09x10^{13}	~2.63x10^{35}	~1.27x10^{89}	∞
total ILs with 3 rules ($2^N \cdot 2^N \cdot N!$)	32	3.84x10^2	~6.14x10^3	~1.23x10^5	~2.95x10^6	…
unrepeated ILs with 3 rules	4 (12.5%)	24 (6.25%)	1.92x10^2 (3.125%)	1.92x10^3 (~1.564%)	2.31x10^4 (~0.782%)	…

3.2.3 Singular Toggles

Another approach was made dealing with minor bit changes. This time it were considered operators (toggles) that provide simple modifications in the bit strings, preserving the univocity.

Considering a bit string with a length of s bits, we have 2^s toggles that deal with every possible combination of bit changes. For instance, T_{011} means the toggle of the first two bits but not of the last one. However, changes simultaneously in more than one bit can be obtained by combinations of single bit toggles (e.g., $T_{11} = T_{01} \cdot T_{10}$). In this sense, to change the bit string in just one bit at a time there are only s different operators (T).

A specific kind of non-trivial toggles I_{ij} was also considered in addition to the single toggles. These toggles regard 2 bits at a time, and operate as follows: if the two bits are equal then these two bits are changed. There are C_2^s different combinations of two bits (C_k^n) and so the same number of the two bit toggles considered. In the opposite way operates D_{ij}: if the two bits are different then these two bits are changed.

These toggles imply correlated bit changes that find motivation in the NK models [20]. In table 3.3, there are present all the operators calculated for two-bit strings. The red columns show operators that can be obtained by a combination of the others.

Table 3.3 Singular toggles considered and the result of their application.

T_{00}	T_{10}	T_{01}	T_{11}	I_{12}	D_{12}
00	10	01	11	11	00
01	11	00	10	01	10
10	00	11	01	10	01
11	01	10	00	00	11

From these relationships is easy to verify that the operator D_{12} is easily obtained by:

$$T_{01}.T_{10}.I_{12} = D_{12} \tag{3.11}$$

Another fact is that the operator D_{12} is equivalent to the exchange between two-bit significances. This corresponds to the operator of exchanging between bit positions ("weights"), mentioned in the *TWR* rule.

In fact, different operations were tested with all the singular toggles considered of table 3.1. Combinations of maximum six operators at a time were tested. Calculations showed that no more unrepeated ILists were possible to generate beyond the number of ILists generated by the *TWR* rule. Consequently, higher diversity could not be achieved.

3.3 Remarks

The mathematical binary rules presented avoid the storage of the full ILists. T cell receptors will use these rules for their receptors to rank APC ligands. A score is calculated for each APC ligand that corresponds to a position in the IList (rank). The rules considered in this chapter are ILists' generator functions.

The *TWR* rule promotes the symmetrical positioning of the pathogens along the possible ILs, because of the univocity. This means, for instance, that pathogens will be positioned in the first places on the total possible ILists the same number of times as in the last places. This may mislead to suggesting that pathogens would have *a priori* the same probability of maximum detection than of maximum non-detection. Nevertheless, it is important to clarify that it takes only one cell that is less frustrated with the pathogen, so that the pathogen is detected. Each detection system consists of a certain number of T cells with different receptors which rank the pathogen differently. Depending on this ranking, pathogens can be differently captured to form a stable conjugate.

The *#rcb* rule provides no symmetrical positioning of pathogens. This is a consequence of the non-univocity. This rule introduces a high degeneracy in the APCs' ranking, given that many APCs will be ranked in the same position.

It was also studied the introduction of a second non-trivial toggle in the *TWR* rule. However, the rules previously considered showed to possess the same diversity in the ILists. This was proved to be a similar case to the singular toggles.

This study showed that binary coding is correlated due to the well-defined bit significances. This fact has the consequence that is not easy to generate very comprehensive and diverse rules relative to the full ILists.

Chapter 4

Cellular Frustration Algorithm

The work developed for this thesis follows the PhD work under development by Patricia Mostardinha. In her PhD, a Cellular Frustration algorithm (CFA) was tested using the Interaction Lists (ILs) formalism (chapters 2 and 3 [11]). The purpose of this thesis is to develop similar nonself detection algorithms but, extending the concept to general data sets and compressing the information contained in the Interaction Lists. The importance of such implementation allows the potential use in practical applications, for instance, related with the problem addressed by Forrest and collaborators. The problem is how to detect an intruder, with no errors, in a potentially large data set and to do this using a computationally time efficient method.

In this chapter the computational algorithm is explained and results from numerical simulations using different parameters and optimizations are analysed. The performance of the algorithm will be compared with the performance of the NSAs discussed in chapter 1.

4.1 Algorithm outline

The immunological concepts that motivated the formulation of the algorithm were explained in chapter 2. It is assumed that specialized antigen presenting cells (e.g., macrophages, dendritic cells…) flow through the body capturing antigens and fragmenting them into antigenic peptides. Pieces of these peptides are assembled in the major histocompatibility complex (MHC) and displayed on the surface of the cell. T cells have receptors that recognize the different peptide-MHC combinations (ligands) on the APCs surface. T cells are activated by nonself APCs, i.e., those presenting foreign (pathogenic) peptides. This typically requires a certain amount of time. It thus requires that an immunological synapse is sustained for a sufficient long time. After activation, T cells go to the

location in the body where peptides were collected and secrete lymphokines or chemical signals, in order to attack pathogens, either directly or through T cell help [5].

T cells can be abstractly viewed as mobile and independent detectors. These cells have no centralized control and little, if any, hierarchical control [4]. They act by interacting through simple and localized rules. These rules provide the basis for the interaction with APCs and therefore to self/nonself discrimination. APCs present information for recognition by T cells. APCs interact with T cells independently of whether they present self or nonself peptides. Hence, it should be assumed that APCs interact with T cells using interaction rules that are independent on peptide classification.

The Cellular Frustration framework assumes that both T cells and APCs have receptors and ligands. Furthermore, a crucial assumption in the theory is that both cell types perform cellular decisions that work as an optimization process: each cell maintains only stable contacts that increase the avidity of signals received by their receptors. This depends on the matching between receptors on one cell and ligands on one other cell type. In this algorithm the information contained in ligands and receptors is stored in binary strings for each cell type.

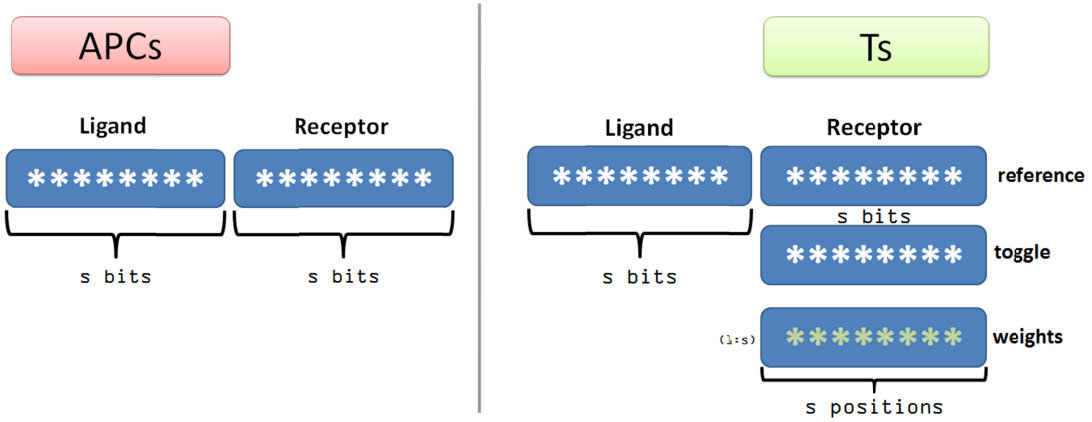

Fig 4.1 Information structure of T cells and APCs used in the Cellular Frustration Algorithm.

For APCs, two strings of s bits are considered: one for the ligand and one for the receptor (fig. 4.1). For simplicity the same string is considered for both, indicating that correlations exist between ligands and receptors. These correlations could take place through intra-cellular processes.

T cells receptors have higher complexity and diversity. To use with the rules explained in chapter 3, T cells receptors consists of two strings of s bits (called "toggle" and "reference") and a string of s positions (called "weights"). The latter string with s positions has integers between $0: s-1$, the bit significances of a s bits binary string. T cell ligands are simpler and consist only of a string of s bits. The information that is stored in all cells ligands and receptors is summarized in figure 4.1.

The dynamics of the system is established by individual cellular decisions. Decisions are based on the rules presented in chapters 2 and 3. If given the opportunity each cell has to decide whether to establish a binding to another cell or not. If it is alone, then it binds to any cell (reactivity is assumed to

be maximal). However, when a cell receives signals from two cells, it prioritizes interactions with the cell delivering the stronger signal. All cells perform this type of decision making. Consequently, if two cells in different conjugates interact, then a new conjugate is formed only if both cells sense stronger signals arising from each other. From a computational point of view, decisions minimize the scores involving the receptor information of the decision-maker cell and the ligands information of the other cells.

APCs receptors rank T cells according to the distance of their ligand string to the APC's receptor string. On the other hand, T cells apply the rules of chapter 3 (TWF or $\#rcb$) to rank APCs. The decision will depend on their receptor strings and the APCs ligand strings.

The algorithm is divided in two phases: *education* and *monitoring*. The process of selection and maturation of T cells (*negative selection*) is replicated in this algorithm and is designated as *education*. The process of positive selection is assumed to have occurred before this process, so that only fully functional T cells are considered.

In this algorithm, only the T cells receptors will be subject to *education* (changed). The receptor strings involved in the computation of the scores (fig. 4.1) will be changed to maximize frustration. This process of education consists on minimizing long-lived interactions with self. T cells that exhibit long lifetimes will have their receptor information changed, as if one T cell would have been eliminated and a new one arrived from the constant flow of incoming cells to the thymus. The performance of the nonself detection is highly dependent on the education process. For the same self set, different T cell receptors information will originate different responses to the same pathogens.

The *monitoring* phase consists on the introduction of pathogens and consequent monitoring of conjugate lifetimes. The simulation of the cellular interactions is the same as in phase I, with the difference that one extra APC (nonself) is added to the system. Pathogens are APCs with strings that are different from the self APCs.

A generic Cellular Frustration algorithm (CFA) is presented in pseudo-code below.

Algorithm 4.1: Generic Cellular Frustration Algorithm.

input: size of bit strings $\boldsymbol{s}$, $\{\boldsymbol{N_{APC}} = \boldsymbol{N_T}\} \in [0, 2^s - 1]$
$random\{\boldsymbol{Lig(APCs)} = \boldsymbol{Rec(APCs)}, \boldsymbol{Lig(Ts)}, \boldsymbol{Rec(Ts)}\} \in [0, 2^s - 1]$
$\boldsymbol{Lig(Pathog)} = \boldsymbol{Rec(Pathog)}$
$\boldsymbol{Lig(APCs)} \subseteq \boldsymbol{S}$, $\boldsymbol{Lig(Pathog)} \subset \boldsymbol{NS}, \boldsymbol{S} \cup \boldsymbol{NS} = \boldsymbol{U}$
$\boldsymbol{Rec(Ts)}\{Tog, W, Ref\}$

Phase I: Education

1 **begin**
2 draw random encounters between all Ts with APCs joining them in conjugates and storing the time of their formation

3 **for** $i = 2, \dots, N_{iterations}$
4 draw random encounters between all elements of the system (single or conjugate)
5 evaluate APCs' decisions for changing interaction or not to new Ts
if an APC is alone, the decision is to interact with the new T
otherwise, Ts are ranked based on the score of their ligands **endif**
6 evaluate Ts' decisions for changing interaction or not to new APCs
if a T is alone, the decision is to interact with the new APC
otherwise. APCs are ranked based on the score of their ligands (rule evaluation) **endif**
7 match new conjugates if and only if two decisions are in agreement storing the conjugates lifetimes and destroying the previous conjugates
8 educate the system by randomly changing the receptor information of Ts with the longest lifetimes observed and separate cells of the conjugates that exist with these Ts
9 **endfor**
10 **end**

Phase II: Monitoring

1 **begin**
2 introduce a pathogen as an extra APC with $\boldsymbol{Lig(Pathog)}$ and $\boldsymbol{Rec(Pathog)}$
3 **for** $i = 2, \dots, N_{iterations}$
4 evaluate steps 4 to 7 of the phase I
5 monitor all conjugate times so that the pathogen can be detected by exhibiting a higher number of long conjugates, on average, than the remaining APCs' system (*self*)
6 **endfor**
7 **end**

4.2 Algorithm specifications

Here I will outline some important details on the numerical implementation.

4.2.1 Cellular decisions

– *APCs decisions*

APCs perform decisions to minimize the score, according to their ligand string, calculated by:

$$score = \left[Lig_{(T)} - Rec_{(APC)}\right] \tag{4.1}$$

with periodic boundary conditions.

– *T cells decisions*

Two types of rules were studied concerning T cell decisions (chapter 3). Each T cell receptor's information is used to calculate the scores according to the following formulas:

- *TWR* rule:

$$score = \left(Lig_{(APC)} \oplus Tog\right)_W - Ref \tag{4.2}$$

- *#rcb* rule:

$$score = f_{\#rcb}\left(\left(Lig_{(APC)} \oplus Tog\right)_W,\ Ref\right) \tag{4.3}$$

4.2.2 Education

T cell receptors are initially randomly drawn. The education process continuously changes T cells receptors to maximize frustration and consequently minimize the maximum contact times with self APCs. The simplest education strategy consists in replacing the receptors of those T cells performing the longer contacts, by other randomly drawn receptors. We confirmed that indeed this increases the frustration. However it never reaches a solution that maximizes frustration. A particular maximally frustrated population would have the following T cell receptor information (obtained based on [11]):

- "Toggle" strings are set to zero.
- "Weights" position strings have the bit significances in the standard order for binary strings.
- "Reference" strings should have the APC's ligand string immediately after their own T cell ligand string and rank in the last place the APCs that have the ligand string equal to their own T cell ligand.

Fig 4.2 Example of the receptor for maximal frustration given the T cell ligand

The process of *education* can be optimized using different strategies. One of the possible approaches consists in slowly tuning modifications in the receptor information, instead of using random and more dramatic modifications. The idea is to perform small modifications in the score of the APCs that produce the longest lifetime. By adapting the receptor information to increase the score of the APC this guarantees that the contact with that APC becomes less stable. For instance, to change the score X by δX, changing the "toggle" string, for one APC's ligand (Lig) the new toggle is obtained as follows:

$$score = (Lig \oplus Tog)_W - Ref = X \tag{4.4}$$

$$X + \delta X = (Lig \oplus Tog\_new)_W - Ref \tag{4.5}$$

$$(X + \delta X + Ref)_{W^{-1}} = Lig \oplus Tog\_new \tag{4.6}$$

$$Tog\_new = Lig \oplus (X + \delta X + Ref)_{W^{-1}} \tag{4.7}$$

In a similar way, scores can also be shifted by δX modifying only the "reference" string, which is obtained by the expression:

$$Ref\_new = (Lig \oplus Tog)_W - (X + \delta X) \tag{4.8}$$

To modify the score by changing the "weights" string, one can simply change two bit significances within the s positions.

This approach consists in specifically changing the rank of one APC in the IList of a T cell. However, given that the rules considered are global, changes in the T cell's receptor information affect not only scores relatively to one APC but the scores of the whole population of APCs. This can reduce frustration towards other APCs. Nevertheless, the system can become more frustrated, on average, because the APCs that promote longer conjugate lifetimes are always being the subject of the receptors modification.

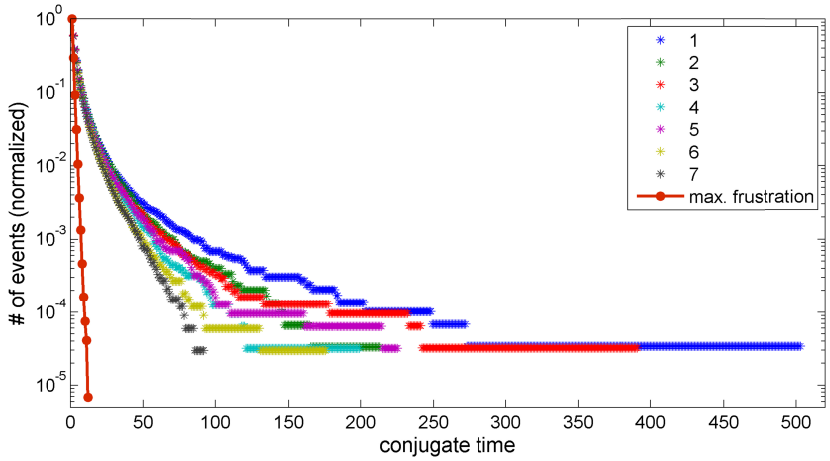

Fig 4.3 Evolution of conjugate lifetimes during the education process using adaptive threshold education. Conjugate lifetimes of the system are shown decreasing towards a maximal frustration regime, shown in red. Consecutive time windows are numbered.

– *Adaptive Threshold Education*

In order to optimize and accelerate the *education* process, adaptive thresholds can be implemented. One of the approaches considered is to change the T cell receptors if a conjugate lifetime overcomes a certain threshold (THS). At the beginning of simulations, long lifetimes are frequent and so THS should be set with a high value so that not most receptors are changed. Afterwards, after some modifications, there are no lifetimes above THS (during a certain time window), and consequently THS is set to the maximum lifetime registered minus one. If only a very small number of conjugates have lifetimes above THS (during the time window), then THS is dropped only by one unit of time. The time window should be large enough to let cells interact and achieve the maximum lifetimes. It should be noticed that THS can never drop below the maximally frustrated conjugate lifetimes. A typical set of histograms for all conjugates lifetimes along the education process is presented in Figure 4.3. It is clear that lifetimes are reduced along the education process.

4.2.3 Monitoring

The *monitoring* process corresponds to the introduction of the pathogenic and analysis of its dynamics and comparison with the dynamics of the other self cells. For the results that we will present single nonself APCs were introduced and tested one by one. These APCs carry ligand and receptor strings not previously seen by T cells ("one-class learning"). The conjugates of the whole system are monitored by counting the number of lifetimes longer than a given threshold. This number is also evaluated for self cells and compared by calculating a ratio between the normalized number of events (the frequency) for self and nonself conjugates.

In figure 4.4 a) is represented the distributions of all lifetime histograms. The conjugate lifetimes for each APC cell are plotted with blue thin lines. The blue thick line highlights the self APC that displays the longest lifetimes (labelled as "worst self"). The average of all self distributions is plotted in black and the pathogen in red. For illustration the joint histogram for a maximally frustrated system is shown in cyan.

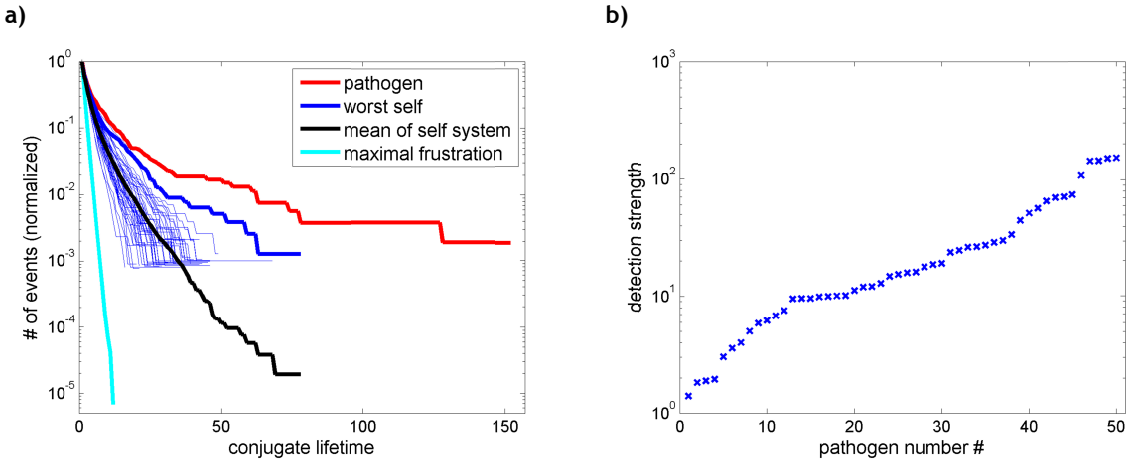

Fig 4.4 Monitoring analysis. a) Lifetimes distributions for a system with $N_{APC} = N_T = 50$ and one pathogen, in the monitoring phase. The maximal frustration regime for the self system is also presented. b) Plot of the detection strength for 50 random pathogens for the ratio using the average of self, demonstrating the variability in detection.

Detection ratios can be evaluated at a given threshold time (for instance, τ=50) using the pathogen histogram and the average of the lifetimes distributions or using the "worst self" histogram. The detection strength is determined by the orders of magnitude of this ratio (the logarithm of this ratio). It is interesting to notice in figure 4.4 a) that if detection was carried out using the maximal frustration regime, the detection strength could potentially be of four orders of magnitude.

A good detection (fig. 4.5 a)) is obtained when the lifetimes distribution of the pathogen is highly separated from the lifetimes distribution of the remaining system, with some orders of magnitude for

long lifetimes. A typical pathogenic distribution of conjugate lifetimes has a higher number of long lifetimes than the distribution for conjugates with self cells.

Non-detection (fig 4.5 b)) occurs when the lifetimes distribution of the remaining system is above the distribution obtained for the pathogen. In this case, the pathogen is as frustrated (or more frustrated) than several self cells.

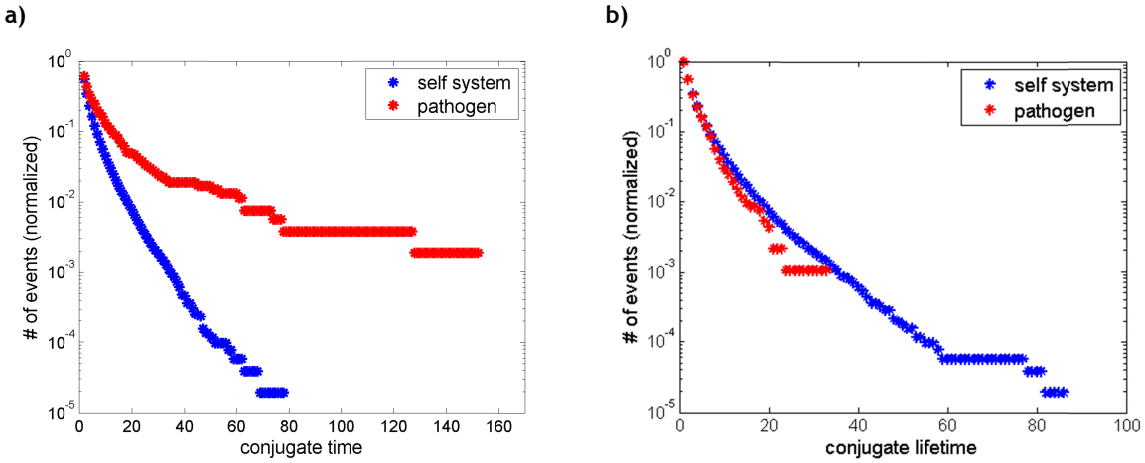

Fig 4.5 Plots showing a good detection with some orders of magnitude (a) and a non-detection (b).

4.3 Results on Self/Nonself detection

The algorithm is suitable for any number of detectors (N_T) or self presenters (N_S). However, this study focused on systems with the same number of detectors and presented strings ($N = N_S = N_T$). All populations were educated for 10^7 iterations and pathogens monitored for 10^4 iterations.

The first set of results is presented in Table 4.1 and uses the same self strings used in chapter 1 (table 1.2). These results show that if in the NSA approach we use the same number of detectors as the size of self, it produces worse results than using Cellular Frustration algorithm (CFA). However it is clear that results using the "worst self" histogram are quite poor. This is due to the large N used.

Table 4.1 Results for known self sets (also used in chapter 1, table 1.2).

N	s bits	Rule	Non-detection		N educated populations	Orders of Magnitude	
			mean	worst self		mean	worst self
100	9	*TWR*	0%	7.77%	10	0 - 2.5	
		#rcb	0%	3.64%		0 - 1	
150		*TWR*	0%	19.89%		0 - 2	
		#rcb	0%	21.82%		0 - 1	

Simulations were also performed exhaustively for different self sizes (table 4.2). For each value of N the set of self strings was fixed and ten detector populations were generated. Good detection is observed for $s = 9$ bits, obtaining better results with the *#rcb* rule. In tables 4.1 and 4.2 simulations

show better results for smaller self sets, approaching perfect detection ($< 1\%$) even when the pathogen behavior is compared to the worst self cell. For $s = 9$ bits, perfect detection is achieved relatively to the average of self. Detection strengths are typically higher for the *TWR* rule.

Table 4.2 Results for 10 random self sets for three different self sizes (standard deviation in brackets).

N	s bits	Rule	Non-detection		N educated populations	Orders of Magnitude	
			mean	worst self		mean	worst self
25	9	*TWR*	0%	1.97% (0.74%)	10	0.5 - 3.75	0 - 2.25
		#rcb	0%	0.27% (0.16%)		0.2 - 2.2	0 - 1.75
50		*TWR*	0%	3.25% (1.08%)		0.5 - 3.5	0 - 2
		#rcb	0%	0.87% (0.35%)		0.2 - 2	0 - 1
100		*TWR*	0%	10.53% (1.97%)		0.5 - 2.5	0 - 1.75
		#rcb	0%	5.33% (1.21%)		0.2 - 1.8	0 - 1

Simulations were also run with $s = 12$. In this case, good detection is obtained with the best results with the *#rcb* rule. Results approach perfect detection when s increases. This contrasts with NSAs which, for a fixed number of detectors, decrease performances when s grows.

Table 4.3 Results for a larger self set ($s = 12$).

N	s bits	Rule	Non-detection		N educated populations	Orders of magnitude	
			mean	worst self		mean	worst self
100	12	*TWR*	0%	5.60%	10	0 - 2	
		#rcb	0%	0.70%		0 - 1.75	

4.4 Final Remarks

In this chapter we developed nonself detection algorithms using the Cellular Frustration framework. These algorithms provide detection of anomalous data (nonself) and could be used for any type of data. Designing practical algorithms is an engineering task that depends on specific strategies like the rules used, and the parameters chosen. This work can still be subject of further optimizations.

The algorithm presented can be classified as an activity monitor, rather than a signature scanner or file authentication program as the approaches by Forrest and collaborators. Nevertheless, the algorithm's purpose is similar to that of NSAs, i.e., to discriminate self and nonself patterns, giving an input of self samples only ("one-class learning").

The Cellular Frustration algorithm uses a minimum number of detectors. The number of detectors is equal to the size of self. This fact establishes a difference with NSA approaches. So far the Cellular Frustration Algorithm would outperform NSA for small N or for large string sizes. In principle perfect nonself detection is achievable using CFA. However, detection performances depend crucially on education process and it also changes with the rule used. Improving algorithms in this direction is still a matter of work in our group.

The *#rcb* rule gave the best results. The degeneracy introduced by this rule may be a useful source of intrinsic frustration. If so, it may avoid requiring extensive education to achieve good results. This "intrinsic frustration" is also useful for increasingly larger systems, for which it is harder to approach maximal frustration.

The worse results were obtained for an increasing size of the self set (tables 4.1 & 4.2). This should also be related to the fact that the diversity is rather restrained, as seen in Chapter 3. Since we used small space sizes, 2^9, the education process is constrained when the number of strings increases. Possibly larger systems would require a larger amount of education to achieve similar results.

On the other hand, increasing the string space size (s), improves detection. These results are still preliminary and would require more extensive simulations and statistical analysis. However, it is clear that for instance, if we had $s = 32$ bits, the Cellular Frustration algorithm would clearly outperform NSAs. In this case, the CFA uses the same number of detectors, and NSAs would require $\sim 10^9$ detectors to achieve 1% failure probabilities.

For this thesis I benefited from the work being developed by Bruno Faria on the same topic. He is improving this work using other strategies. I had developed simulations using MATLAB. However, MATLAB is very inefficient in what concerns computational speed. Bruno rewrote CFAs using C programming. Computational time for 10^4 iterations in the monitoring process was about 150 times larger in MATLAB than in C. For this reason some of the results in this chapter were obtained with Bruno's program, for which I much acknowledge him. The benchmark was performed in a Core i7-860 (2.8 GHz) with 8Gb RAM.

Conclusions

The Cellular Frustration framework is a novel mechanism completely different from previous theories on self/nonself discrimination. The question posed was to know if this mechanism could solve a problem of self/nonself discrimination as addressed by other methods existent in the literature. Although conceptually there was already that possibility, the challenge of this thesis was to develop algorithms susceptible of practical applications.

Implementation issues were identified which makes non-trivial the passage between the conceptual framework to the practical implementation. For instance, how the information is coded at the receptor level implies information compression that influences the results.

On the other hand, different implementation strategies were identified, both at the level of the computational parameters and at the level of the distances between strings that were considered.

The studies here presented are a preliminary attempt in obtaining innovative algorithms for intrusion detection. It was demonstrated that the method of Cellular Frustration algorithm (CFA) can work in achieving almost perfect detection. This result is non-trivial given that it is not trivial that the Cellular Frustration framework works for this problem. The Cellular Frustration framework is based on assumptions much less trivial than the assumptions considered in NSAs. However, the algorithms here proposed are still not accurate when the number of self strings is large.

References

[1] Forrest, S. et al. (1994) "Self-Nonself Discrimination in a Computer", in *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*. IEEE Press.

[2] D'haeseleer, P. et al. (1996) "An immunological approach to change-detection: algorithms, analysis and implications", in *Proceedings of the 1996 IEEE Symposium on Computer Security and Privacy*. IEEE Press.

[3] D'haeseleer, P. et al. (1997) "A Distributed Approach to Anomaly Detection", submitted to *ACM Trans. Inf. Sys. Sec.*.

[4] Hofmeyr, S. (1999) "An Immunological Model of Distributed Detection and Its Application to Computer Security", PhD Thesis, Dept of Computer Science, University of New Mexico.

[5] Dasgupta, D. and Niño, L.F. (2009) "Immunological computation: theory and applications", Auerback Publications.

[6] Timmis, J. et al. (2008) "Theoretical advances in artificial immune systems", *Theor. Comp. Sci.* **403**, 11-32.

[7] Chan, C. et al. (2003) "T cell sensitivity and specificity – kinetic proofreading revisited", *Discr. Cont. Dyn. Sys.B* **3**, 343-360.

[8] Harmer, P.K., et al. (2002) "An Artificial Immune System Architecture for Computer Security Applications", *IEEE Trans. Evol. Comp.* **6**(3), 252-280.

[9] Kim, J. and Bentley, P. (1999) "An Evaluation of Negative Selection in an Artificial Immune System for Network Intrusion Detection", in L. Spector et al. (eds.): *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 1330-1337.

[10] Abreu, F.V., et al. (2006) "Cellular Frustration: A New Conceptual Framework for Understanding Cell-Mediated Immune Responses", *LNCS* **4163**, 37-51.

[11] Abreu, F.V. and Mostardinha, P. (2008) "Maximal frustration as an immunological principle", *J. R. Soc. Interface* **6**(32), 321-334.

[12] Greensmith, J., et al. (2010) "Artificial Immune Systems", in Gendreau, M. and Potvin, J.-Y.:"Handbook of Metaheuristics", Springer, 421-448.

[13] Valitutti, S., et al. (2010) "The space and time frames of T cell activation at the immunological synapse", *FEBS Lett.*, doi:10.1016/j.febslet.2010.10.010 .

[14] Hopfield, J.J. (1974) "Kinetic Proofreading: A New Mechanism for Reducing Errors in Biosynthetic Processes Requiring High Specificity", *Proc. Natl. Acad. Sci. USA* **71**(10), 4135-4139.

[15] McKeithan, T.W. (1995) "Kinetic proofreading in T-cell receptor signal transduction", *Proc. Natl. Acad. Sci. USA* **92**, 5042-5046.

[16] Depoil, D., et al. (2005) "Immunological synapses are versatile structures enabling selective T cell polarization", *Immunity* **22**(2), 185-194.

[17] Binder, P.-M. (2008) "Frustration in Complexity", *Science* **320**, 322-323.

[18] Almeida, C.R. and de Abreu, F.V. (2003) "Dynamical instabilities lead to sympatric speciation", *Evol. Ecol. Res.* **5**(5), 739-757.

[19] Lindo, A.M. (2010) "Molecular Frustration model with tunable specificity", BSc Thesis, Dept. of Physics, Aveiro University.

[20] Kauffman, S. (1996) "At Home in the Universe: The Search for the Laws of Self-Organization and Complexity", Oxford University Press.